UNIVERSIDAD DE ALCALÁ
Escuela Técnica Superior de Ingeniería Informática

GRADO EN INGENIERÍA DE COMPUTADORES

Trabajo Fin de Grado

"**HERMES FORUM**"

Aplicación iOS para la creación y gestión de foros de forma instantánea.

Autor: José Antonio Murillo Martín

Director/es: Juan José Sánchez Peña

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

CALIFICACIÓN:
FECHA:

AGRADECIMIENTOS

El presente trabajo fue realizado bajo la supervisión de Juan José Sánchez Peña, a quien me gustaría expresar mi más profundo agradecimiento, por la oportunidad de realizar este proyecto.

A mi madre, por ser el apoyo más grande durante mi educación universitaria, porque sin tu esfuerzo y compresión no habría sido posible llegar donde estoy.

A mi familia, por el apoyo que siempre me brindaron en el transcurso de mi carrera, en especial a mis sobrinas, que aportaron felicidad en los momentos sombríos.

A mis compañeros y amigos, que me acompañaron durante la carrera y me prestaron su ayuda, en especial a Raúl, Eduardo e Irene, gracias por confiar en mí y no dejarme desfallecer.

Y por último, a todos los profesores, por compartir sus conocimientos conmigo y enseñarme el camino.

A todos ellos,

Muchas gracias.

ÍNDICE GENERAL

Contenido

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ACRÓNIMOS

-A-	
API	Application Programming Interface
App	Application software
-B-	
BaaS	Backend as a service
BBS	Bulletin Board System
-G-	
GCC	GNU Compiler Collection
GCM	Google Cloud Messaging
GNU	GNU´s Not UNIX
GUI	Graphical User Interface
-I-	
IDE	Integrated Development Environment
IOS	iPhone Operative System
-L-	
LLVM	Low Level Virtual Machine
-M-	
Malware	Malicious software
-O-	
OMT	Organización Mundial del Turismo
-S-	
SDK	Software Development Kit
-T-	
TCP	Transmission Control Protocol
-U-	
UDID	Unique Device Identifier
UDP	User Datagram Protocol
-V-	
VoIP	Voice Over IP

RESUMEN

"**HERMES FORUM**", APLICACIÓN IOS PARA LA CREACIÓN Y GESTIÓN DE FOROS DE FORMA INSTANTÁNEA

Autor: **Murillo Martín, José Antonio**
Director: Juan José Sánchez Peña

RESUMEN DEL PROYECTO

El volumen de turistas internacionales fue de 1.138 millones en 2014, 51 millones más que en 2013. Con el incremento del 4,7 %, este es el quinto año consecutivo que se supera la media desde la crisis económica de 2009.
Cuando una persona realiza un viaje sin motivo de lucro, requiere de una innumerable cantidad de servicios que van desde transporte, hospedaje y alimentos hasta compras, distracciones, o esparcimiento. Así, el turismo engloba un conjunto de actividades que producen los bienes y servicios que demandan los turistas.
Gracias a estos resultados, aumenta la importancia de la información y la planificación en los viajes.
El proyecto se concentrará en aprovechar esa necesidad de servicios, desarrollando una aplicación iOS, para la creación y gestión de foros, que pueda ayudarnos a solucionar todas las dificultades que surjan a la hora de planificar un viaje.

ABSTRACT

The volume of international tourists was 1,138,000 in 2014, 51 million more than in 2013. With the increase of 4.7%, this is the fifth consecutive year that exceeds the average since the economic crisis of 2009.
When a person makes a trip without profit motive, it requires a myriad of services ranging from transportation, lodging and food to shopping, entertainment or recreation. Thus, tourism comprises a set of activities which produce goods and services demanded by tourists.
Thanks to these results, there is an increase in the importance of information and planning on trips.
The project will focus on the need to take advantage of services, developing an iOS application, for the creation and management of forums, which can help us solve all the difficulties that arise when planning a trip.

1. INTRODUCCIÓN

Para poder entender la idea originaria de este proyecto, vamos a intentar proporcionar al lector una visión general de la misma.

¿Qué es un Foro?

Son los sucesores modernos de los sistemas de noticias BBS y Usenet, muy populares en los años 1980 y 1990. Los foros son una herramienta que permite establecer contacto con otros usuarios de internet y generar comunicación sobre temas diversos. También funcionan como una importante fuente de información a la hora de realizar consultas y buscar asesoramiento de primera mano.

El objetivo del foro es conocer las opiniones sobre un tema concreto. Comúnmente, un foro en internet permite que el administrador del sitio defina varios temas sobre una sola plataforma, que funcionarán como contenedores de las discusiones que comenzarán los usuarios y donde otros podrán responder o empezar un nuevo debate.

¿Cuáles son sus ventajas?

- Se comparten opiniones, experiencias y dudas sobre un tema.
- Expresar y responder opiniones.
- Se conocen opiniones de diferentes personas para un mismo tema
- Aunque uno entre después se puede entender el tema.
- Favorece el desarrollo de habilidades sociales mediante la interacción y la empatía, permitiendo un interaprendizaje.
- Permite el desarrollo del pensamiento crítico desde el compartir opiniones, y sacar conclusiones.
- Ayuda a mejorar las habilidades de comunicación escrita a partir de los debates, comentarios, análisis y crítica de textos; así mismo desde la discusión, conclusiones, resolución y comparación de casos.

En este proyecto, intentaremos aprovechar todas las ventajas ofrecidas por los foros, diseñando una aplicación iOS para la creación y gestión de foros de forma instantánea. Al ser desarrollado en una plataforma móvil, conseguiremos una mayor libertad y un contacto entre usuarios más rápido e intuitivo a través de Internet.

Aprovechando las capacidades del sistema operativo iOS, podremos añadir varias funcionalidades impensables en un foro, como poder realizar llamadas, chatear con amigos, subir fotos, enviar mensajes y mucho más.

Además, el atractivo personal de este proyecto es poder aprender a utilizar el lenguaje de programación objective-c, que junto a xcode, posibilita las herramientas necesarias para desarrollar una aplicación en el sistema operativo iOS de Apple.

1.1 CONTEXTO

Este proyecto se centrará inicialmente en el sector turístico, por ser el que más se ha incrementado en los últimos años, ofreciendo una mayor demanda de servicios e información. El proyecto podrá alojar, más adelante, otros sectores como deportes, finanzas, educación, etc.

¿Por qué se ha desarrollado tanto el turismo en los últimos tiempos? Diversas causas explican este fenómeno:

- La mejora de los transportes, cada vez más rápidos y baratos.
- El incremento y la mejora de las infraestructuras turísticas:
    - hoteles, apartamentos, restaurantes, comercios, centros de ocio…
- El aumento del nivel cultural y el nivel de vida, que ha elevado la curiosidad por conocer monumentos, lugares, culturas…
- El desarrollo de la industria turística capaz de atraer a millones de personas mediante la publicidad sistemática.

También hay que añadir la ayuda de las redes sociales, que han supuesto un gran avance en cuanto a la comunicación global. La capacidad de transmitir cualquier información de forma instantánea a cualquier parte del mundo ha abierto puertas a muchas empresas y sectores, uno de los cuales ha sido el sector turístico.

Las personas de hoy, con una cuenta en cualquier red social, actúan de forma que cuando se van de viaje, se conectan a la red en todo momento posible para compartir sus experiencias y opiniones.

Si queremos aprovechar estos factores, será necesario desarrollar una aplicación que pueda ofrecer las mismas posibilidades que una red social, permitiendo comunicarse libremente a todos los usuarios.

Para poder hacer un buen uso de la aplicación, seguiremos unos claros objetivos.

- Hacer que la aplicación sea visualmente atractiva. Este punto es importante ya que si queremos que la gente se fije en nosotros es necesario un foro llamativo e interesante para el usuario.
- Tendremos que tener claro a qué público queremos captar o a quien puede interesarle nuestra aplicación.
- Tener la posibilidad de comunicarse libremente, no solo a través del foro, sino mediante mensajes, chat, etc.
- Permitir la posibilidad de buscar amigos o invitar a desconocidos. De esta forma añadimos otra de las cualidades implícitas en las redes sociales.
- Finalmente, quizá el objetivo más importante es la de ser constante, mantener el foro activo y que siempre haya un flujo de gente que lo visite y utilice. Esto se

consigue cumpliendo con los puntos anteriores, colgando fotos, anuncios, promociones, etc.

Las redes sociales son una herramienta muy poderosa, como ya hemos visto, la gente y las cuentas de sus redes sociales son una misma persona. Detrás de cada seguidor que podamos conseguir, hay otra persona, un posible futuro usuario, por lo tanto podemos sacar la conclusión de que no le depara nada malo a nuestro proyecto, si somos capaces de implementar todos nuestros objetivos.

1.2 Motivación

Mi idea original era realizar una App para dispositivos con el sistema operativo iOS. Quería conocer la afamada herramienta Xcode y unirme a la moda de desarrollar una App, pero cuando comencé este proyecto, no tenía experiencia previa en el desarrollo de aplicaciones móviles o con el lenguaje objective-c.

Pretendía con el proyecto, conseguir nuevos conocimientos y experiencia, que me ayudarán en la búsqueda de un trabajo y que me permitiera seguir formándome como profesional de la informática.

Así fue como en la búsqueda de un TFG de estas características, me encontré con la propuesta de Juan José Sánchez Peña. Teniendo la idea de realizar una aplicación de gestión de foros, que podría tener infinidad de posibilidades futuras.

La realización de este proyecto, surge de una necesidad muy real. Cada vez más gente sueña con hacer su viaje ideal, pero son tantas las opciones y las posibilidades, que se hace necesario un guía. Por eso, no podríamos haber elegido un momento mejor para la creación de este proyecto. Con la ayuda de mi aplicación, será mucho más fácil la planificación de un viaje, recibiendo consejos y sugerencias de otros usuarios que han pasado por lo mismo.

Así fue como nació **Hermes Forum**, una aplicación para iOS que nos permite crear y gestionar diferentes foros de una forma sencilla e intuitiva. Haciendo la vida más fácil a sus usuarios y permitiendo además, añadir el plus de pertenecer a una red social en pleno crecimiento.

Desde la aparición de las **redes sociales**, se ha comenzado a diseñar de forma diferente la realización de los viajes. Internet, por no hablar de las redes sociales, son un invento relativamente nuevo. En muy poco tiempo han evolucionado y se han convertido en algo necesario para casi cualquier ser humano del primer mundo, se han convertido en el medio de difusión más grande que existe, y en el que el protagonista eres tú, el mismo usuario, ya que posibilita la interacción con los usuarios estableciendo un vínculo relacional virtual.

1.3 ESTRUCTURA DEL DOCUMENTO

El contenido de esta memoria está estructurado de la siguiente manera:

CAPÍTULO 1 – INTRODUCCIÓN.

En este capítulo se establece el contexto que abarca el desarrollo del proyecto, narrando brevemente las ideas del mismo y presentando las motivaciones y soluciones para satisfacer las necesidades del problema planteado.

CAPÍTULO 2 – JUSTIFICACIÓN DEL PROYECTO.

En este capítulo explicaremos las razones que nos llevaron a idear la aplicación, los objetivos de usuario que nos hemos propuesto y las herramientas necesarias para poder llevarla a buen fin.

CAPÍTULO 3 – ANÁLISIS DE REQUISITOS.

Aquí se recoge el proceso utilizado para conseguir los requisitos funcionales y no funcionales. También se describe los dos servicios que se utilizarán en la siguiente etapa del proyecto.

CAPÍTULO 4 – DISEÑO E IMPLEMENTACIÓN.

Este capítulo está dedicado a la fase de diseño e implementación. Basándose en las necesidades del sistema y la información de la etapa de análisis. Se explicarán los pasos seguidos para su desarrollo y el manejo de los servicios C2Call y Parse, utilizados para implementar la mayoría de los requisitos.

CAPÍTULO 5 – PRUEBAS.

En este capítulo explicaremos la estrategia utilizada para localizar todos los errores, las pruebas realizadas para asegurar que la aplicación cumple los requisitos y todas las soluciones planteadas para su arreglo.

CAPÍTULO 6 – MANUAL DE USUARIO.

En este capítulo se describe el funcionamiento completo de la aplicación. Explicando de forma resumida y en un lenguaje sencillo, todos los pasos a seguir para utilizar la aplicación "Hermes Forum".

CAPÍTULO 7 – CONCLUSIONES.

Este capítulo recoge las conclusiones extraídas a lo largo del desarrollo del proyecto, basándose en los objetivos iniciales y el proceso completo de realización.

CAPÍTULO 8 – TRABAJOS FUTUROS.

En este último capítulo, explicaremos las mejoras que pueden ser implementadas en el futuro, así como las posibles funcionalidades que se podrían añadir y algunos aspectos a tener en cuenta.

ANEXOS

En este apartado se incluirá:

- La planificación, donde se detallan todos los pasos seguidos y el tiempo utilizado para el desarrollo del proyecto.
- El presupuesto, donde se recoge la estimación económica de la app, teniendo en cuenta el número de horas trabajadas, la adquisición de licencias, el material utilizado y demás costes asociados.
- La Bibliografía, donde se detallarán las consultas bibliográficas utilizadas para la realización del documento.

2. JUSTIFICACIÓN DEL PROYECTO

Como se menciona en el resumen, el turismo subirá en Europa un 4%, siendo ya la región más visitada con un total de 588 millones. Gracias a esos resultados, aumenta la importancia de la información y la planificación en los viajes.

Aunque mucha gente piense que los foros están algo muertos y que suenan a algo del siglo pasado, lo cierto es que siguen existiendo **muchos foros interesantes** a los que se podría acudir para buscar información.
Sin embargo, en el mundo del turismo, los foros han sido desplazados poco a poco por **redes sociales** como Facebook o Twitter, donde los usuarios tienen una mayor oportunidad de compartir su opinión, discutir sobre lugares de interés, incluso de conectarse con otro grupo de personas para hablar sobre el destino elegido, medio de transporte a utilizar, atractivos de cada lugar, propósito del viaje, etc.
Asimismo, el sector turístico se ha caracterizado por tener una gran variedad de posibilidades en línea, hoy en día puedes encontrar buscadores por Internet para hoteles, vuelos, restaurantes, zonas turísticas, etc.

Pero, ¿qué ocurre si algo falla en el viaje?, la mayoría de los amigos que conocemos en las redes sociales viven en nuestra comunidad y país, ¿sabrían como ayudarnos?, ¿y qué ocurriría con los portales de viajes?, ¿se harían cargo de problemas que no estuvieran relacionados con hoteles o vuelos?

¿Cómo estar preparados para estas eventualidades?

La solución podría ser un sencillo gestor de foros de viajes, donde se pudiera conocer las experiencias de estos usuarios, pedirles opiniones y aprender de sus errores.
A pesar de ello, en Apple existen muy pocas aplicaciones que te permitan consultar los foros y todas son a través de Safari.

Consultar un foro desde Safari en el iPhone o iPad no es nada incómodo, pero tiene dos desventajas.

- La primera es que el foro debe de ser compatible con el sistema, cuando visites un foro desde el iPhone te indicará si es compatible o no.
- La segunda es más problemática, por omisión, Safari muestra determinadas prestaciones web como películas, animaciones y aplicaciones web, que pueden ralentizar mucho la conexión o permitir ataques de **Malware**. Para evitar muchos problemas, tendrás que desactivar algunas de estas prestaciones para ayudar a proteger tu privacidad y tu dispositivo ante posibles riesgos para la seguridad en Internet.

Para ello ha nacido **Hermes Forum**, una aplicación nativa para iOS, pensada para crear y leer foros de forma mucho más cómoda desde nuestro dispositivo.

Las mismas opciones que podríamos realizar desde el foro directamente pero sin necesidad de pelearnos con una interfaz algo incómoda para ser usada desde un iPhone. Un sencillo gestor de temas de viajes, donde se podrá conocer las experiencias de otros usuarios, pedir opiniones y consejos, buscar amigos, conocer gente nueva y sobre todo, ayudar con esos problemas ocasionales.

Todo ello, de una forma sencilla e intuitiva, con una aplicación que nos permite además chatear, subir fotos o vídeos, hacer llamadas **VoIP**, vídeo llamadas en grupo, y mucho más.

2.1 OBJETIVOS

Irse de vacaciones debe ser un momento de desconexión de nuestra vida diaria. Es la oportunidad de descubrir, visitar y probar cosas nuevas. Pero para muchos, el mero hecho de pensar en preparar un viaje puede provocarles el efecto contrario.
Cuando estamos en casa preparando un viaje a algún país lejano, todo puede parecer un poco complicado. Nos entran miedos y nervios en parte, porque sentimos que no tenemos suficiente información. Lo mejor es poder conocer de primera mano la experiencia de alguien que haya viajado por el país al que nos dirigimos, por eso los foros de viajeros pueden ser muy útiles.

El objetivo a conseguir con el desarrollo de esta aplicación es crear una buena herramienta de gestión de temas, donde un usuario podrá conocer las experiencias de otros viajeros y hacer preguntas concretas para su próxima aventura. De esta forma, la aplicación le ayudará a planificar con antelación un viaje, donde otros compañeros y amigos le darán consejos, opiniones e información, consiguiendo un contacto entre usuarios rápido e intuitivo a través de Internet.

La idea principal es la siguiente, cualquier usuario que se descargue la aplicación tendrá tres elementos básicos a su alcance:

1. El primer elemento es el foro típico, donde el usuario podrá dejar comentarios, elegir sus temas de interés o crear algunos nuevos sin la necesidad de invitar a gente o crear grupos. Cada vez que se introduzca un tema, todos los usuarios que tengan instalada la aplicación recibirán una notificación. Si a un usuario le interesa el tema, puede unirse a la conversación.

2. El segundo elemento es la opción de chat, donde podrás chatear y crear grupos de amigos. Para ello, será necesario añadir los contactos en tu agenda. Si ya conoces el email de un amigo, y este ha sido registrado en la aplicación, solo tendrás que utilizar nuestro buscador y el nuevo contacto aparecerá en tu agenda. Además, podrás subir fotos, vídeos, localizaciones y mucho más.

3. El tercer elemento es la opción de conseguir amigos a través de la mensajería instantánea. Podrás enviar invitaciones, que el usuario recibirá en su correo

electrónico, de esta forma evitamos posibles errores si el destinatario está desconectado. Si estos aceptan, tendrás la opción de poder llamarlos, enviarles mensajes o chatear con ellos.

Después estaría la opción de pago mediante el uso de "**in-app-purchase**", que consiste en la adquisición de nuevas herramientas y funcionalidades. El usuario podrá obtener la posibilidad de hacer llamadas de teléfono vía Ip (**VoIP Call**) gratuitas, vídeo llamadas (también gratuitas) y el uso de **C2Call FreePhone** (Llamadas gratis a teléfonos fijos y teléfonos móviles en todo el mundo, gracias a los créditos libres que se pueden obtener en el muro de ofertas al ver vídeos, descargar aplicaciones, etc.)

Todas estas funciones estarán disponibles para cada usuario, facilitando enormemente la comunicación y ayudando a crear una red de contactos que podrá enseñarnos todo lo necesario para la planificación de nuestros viajes.

2.2 ENTORNO DE TRABAJO

Todo sistema operativo, ya sea móvil o de sobremesa, necesita aplicaciones para ser útil y esas aplicaciones se deben desarrollar siguiendo unas pautas y convenciones impuestas por el mismo sistema y hardware en el que se vaya a ejecutar.

Xcode es el entorno de desarrollo integrado (**IDE**) de Apple, creado para trabajar de manera conjunta con iOS. Se encargará de proporcionar el **iOS SDK**, en el cual disponemos de todas las herramientas, compiladores y frameworks necesarios. Se puede descargar de manera gratuita desde la Mac App Store.

IOS es el sistema operativo que da vida a dispositivos como el iPhone, el iPad, el iPod Touch o el Apple TV. Su simplicidad y optimización son sus pilares para que millones de usuarios se decanten por iOS en lugar de escoger otras plataformas que necesitan más hardware para mover con fluidez el sistema. Cuenta con actualizaciones periódicas que están disponibles para su descarga y actualización a través de **iTunes**.

Xcode ofrece a los desarrolladores todo lo necesario para crear aplicaciones para Mac, iPad e iPhone. Entre otras cosas, cuenta con una amplia variedad de herramientas innovadoras como la colección de compiladores del proyecto GNU (GCC), que puede compilar código C, C++, Objective-C, Objective-C++, AppleScript y Java, gracias a una gran gama de modelos de programación.
El editor profesional, que se mantiene enfocado en el código, se ha simplificado para hacer que sea mucho más rápido y fácil de usar, creando un panel secundario que muestra automáticamente los archivos que son más útiles, basándose en el código que se está editando.

La última versión, cuenta con un diseño de interfaz de usuario que unifica la codificación, pruebas y depuración dentro de una única ventana.

Además, el compilador integrado de Apple **LLVM** destaca los errores de codificación a medida que se escribe el código, y a veces, es lo suficientemente inteligente como para solucionar los problemas de forma automática.

2.2.1 OBJECTIVE-C

Objective-C es un lenguaje de programación orientado a objetos, nacido en la década de los 80 y que actualmente se usa como lenguaje principal de programación en Mac OS X, iOS y GNUstep.
Fue creado como un **superconjunto de C**, en otras palabras, es posible compilar cualquier programa escrito en C con un compilador de Objective-C, y también se puede incluir libremente código en C dentro de una clase de Objective-C.
Por eso, la sintaxis y características de C están presentes en Objective-C:

- Sentencias de control de flujo (if, for, while...)
- Tipos de datos fundamentales, estructuras y punteros.
- Conversiones implícitas y explícitas entre tipos.
- El ámbito de las variables: Globales, estáticas o locales.
- Las funciones y su sintaxis.
- Las directivas del preprocesador (añadiendo Objective-C las suyas, así como las llamadas directivas del compilador).

¿Y entonces en qué se diferencian C++ y Objective-C? Pues en muchísimas cosas, empezando por lo obvio: las influencias procedentes de las ideas de Smalltalk, lo cual hace de Objective-C un lenguaje muy limpio, pequeño y por ende, mucho más rápido y fácil de aprender que C++. Aún así, Objective-C es mucho menos usado que C++ (hablando en general, en el mundo Mac OS X no es así); esto puede ser porque, a diferencia de otros lenguajes de las GCC, Objective-C no ha sido estandarizado por ningún organismo internacional, sino que fue NeXSTEP, y ahora Apple con Mac OS X, quienes han contribuido a crear este lenguaje.

Se puede programar en Objective-C tanto en Linux como en Windows, claro está, que el número de librerías es limitado y sólo están las básicas (aunque se pueden acceder a las librerías C y C++ desde Objactive-C).
Después de la diferencia más obvia como es la sintaxis, si se tuviese que elegir una característica que diferencie a Objective-C de otros lenguajes (ya no sólo de C++), ésta sería su dinamismo, en el sentido de que Objective-C es un lenguaje marcadamente dinámico. Muchas de las decisiones que otros lenguajes toman en tiempo de compilación, Objective-C las toma en tiempo de ejecución. Ejemplos:

- En Objective-C, a diferencia de C++, los objetos siempre se crean en memoria dinámica.
- Los atributos de una clase no tienen que estar tipificados estáticamente.
- La comprobación de la existencia de los métodos se lleva a cabo en tiempo de ejecución. ¿Esto qué quiere decir? Pues, que si por ejemplo, llamásemos al

método “imprimirHola” de nuestra clase “Saludar”, pero este no existe, el error nos saltaría durante la ejecución del programa y no durante la compilación.

Este dinamismo puede ser un arma de doble filo, por supuesto. Si por algún casual el método “imprimirHola” si existe, pero nos equivocamos al escribir y al llamarlo escribimos “imprimiHola” (nos comemos la “r” final de imprimir) esto ocasionaría descubrir el error durante la ejecución del programa, algo que en programas de gran envergadura no nos podemos permitir. Por este motivo, durante la compilación del programa, en este caso, nos saltará un “warning” indicándonos que el método que estamos invocando podría no existir, pero el programa compilará y ejecutará sin problemas (si el método no es una rutina crítica para el funcionamiento del programa, claro).

Ejecutando la aplicación

Objective-C es un lenguaje que debe ser compilado. A diferencia de Javascript, PHP, Ruby o Python, que son lenguajes interpretados.
Para compilar podemos seleccionar ***Product > Build*** o presionar ***Command+B*** que es el atajo. Si no dañamos nada del proyecto que generó Xcode para nosotros, debería decir "Build Succeeded".
Si queremos ejecutar la aplicación, seleccionamos el botón de la esquina superior izquierda o podemos también ir a ***Product > Run***, o presionar el atajo ***Command+R***. Xcode compila y corre el programa (no es necesario compilar y ejecutar manualmente, xcode compila siempre antes de ejecutar).

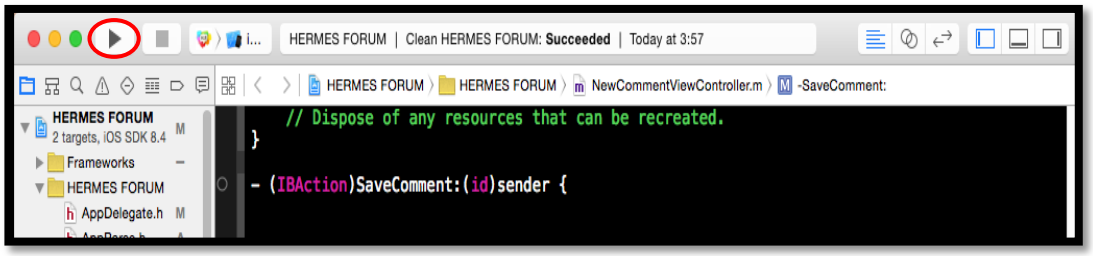


Ilustración 1. Botón ejecutar de xcode.

2.3 BACK-END Y FRONT-END

Para desarrollar este proyecto también fue necesaria la planificación de dos factores muy importantes en la creación de aplicaciones, el Back-end y el Front-end.

A veces nuestras Apps necesitan conectarse con un servidor remoto, para guardar datos, para recibir datos o archivos, o incluso para hacer login e identificarte en la App.
Para que nuestra aplicación haga este tipo de cosas, necesitamos montar un Back-end en un servidor, de forma que la App en cuestión se conecte con este servidor remoto y se puedan ejecutar este tipo de operaciones. Para superar este primer paso usaremos el servicio Parse.

Parse es lo que se denomina un **Back-end as a service (BaaS)** que nos va a permitir disfrutar de múltiples funcionalidades en la nube, permitiéndonos el desarrollo de aplicaciones que requieran un back-end con muy poco esfuerzo.

Los servicios que ofrece Parse principalmente son:

- **Modelo de datos en la nube:** creación de tablas no-SQL en la nube y capacidades para inserción, modificación y consulta vía API.
- **Notificaciones Push:** posibilidad de envío de notificaciones push a nuestros usuarios, previa aceptación por parte del usuario.
- **Cloud Code:** capacidades para la ejecución de código en el servidor, muy útil para la realización de validaciones de seguridad, o procesos automáticos por cambios en los datos.

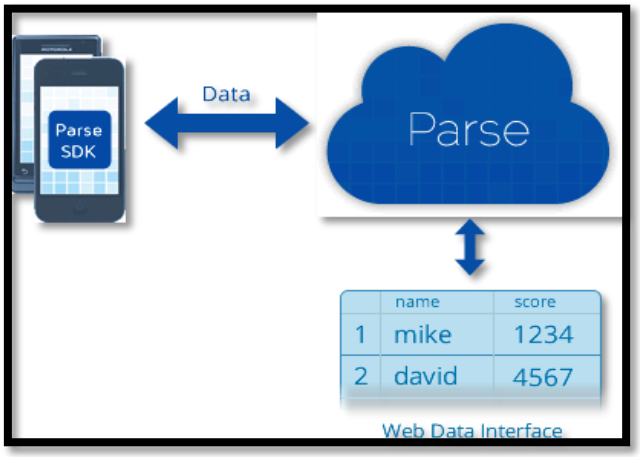


Ilustración 2. Web Data Interface.

Front-end es la parte del software que interactúa con los usuarios, en pocas palabras el diseño de la aplicación, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos. Para superar este segundo paso usaremos los servicios que nos ofrece C2Call.

C2Call es una plataforma que ofrece todos los servicios de comunicación disponibles para nuestra app. Mediante la integración de la tecnología de C2Call y el uso de sus API en las aplicaciones móviles, los desarrolladores pueden tener a todos sus usuarios comunicados con muy poco esfuerzo.

La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y los transforma ajustándolos a las especificaciones que demanda el back-end para poder procesarlos, devolviendo generalmente una respuesta que el front-end recibe y expone al usuario de una forma entendible para este.
Estos dos grandes servicios los veremos con más detalle en el siguiente punto, donde comentaremos lo que nos ofrece cada uno, así como todas sus ventajas y desventajas.

3. ANÁLISIS DE REQUISITOS

Para construir algo primero debe entenderse lo que debe ser ese algo. El proceso de entender y documentar una aplicación software se llama "Análisis de requisitos".
El propósito de la etapa de análisis, es conocer más acerca del contexto en el cual se piensa trabajar.
Esto implica recopilar, organizar y sintetizar la información del contexto del proyecto, para establecer las restricciones que debe cumplir el software, marcar las prioridades que deseamos para nuestra aplicación y buscar los factores que pueden afectar al desarrollo de esta.

Este grado de detalle, tiene como único objetivo eliminar posibles sorpresas durante la ejecución del proyecto y garantizar la claridad de los trabajos a realizar en la fase de desarrollo. Cuando estos trabajos se encuentran claramente definidos, es posible crear una temporización definitiva y garantizar que las sorpresas económicas serán 0.

Al inicio de esta documentación, hemos analizado los requisitos de usuario, precisando las necesidades que deseamos cubrir y los objetivos a cumplir con la aplicación. Ahora describiremos los requisitos funcionales y no funcionales, siguiendo el proceso de reunión de requisitos, que se centra especialmente en la aplicación.

Dentro del proceso de análisis es fundamental que a través de una colección de requerimientos funcionales y no funcionales, el desarrollador de la app comprenda completamente la naturaleza de los programas que deben construirse para desarrollar la aplicación.

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas. En algunas circunstancias, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

Los principales requisitos funcionales están divididos en módulos, que corresponden a las diferentes secciones que componen la App:

1. Login / Registro
    - Crear / acceder a una cuenta de usuario.
    - Autenticación de datos del usuario.
    - Recuperación de contraseña de acceso.
2. Favoritos
    - Listado de los amigos encontrados.
    - Información de los amigos.
    - Listado de los grupos creados.
    - Información de los grupos.
    - Buscador.

3. Favoritos (**Versión Avanzada**)
    - Llamadas VoIP.
    - Vídeo Llamadas.
    - FreePhone.
4. Chat
    - Enviar / descargar mensajes de voz.
    - Enviar / descargar fotos.
    - Enviar / descargar localizaciones.
    - Enviar / descargar vídeos.
    - Crear / enviar mensajes de chat.
5. Foro
    - Crear tema nuevo.
    - Acceder a las normas del foro.
    - Leer comentarios existentes.
    - Crear un comentario nuevo.
    - Invitar a un desconocido a tu lista de amigos.
6. Recientes
    - Mostrar notificaciones push.
    - Mostrar los últimos mensajes del chat.
    - Crear / enviar un mensaje de chat.
7. Grupos
    - Crear / acceder a un grupo.
    - Modificar / eliminar un grupo ya creado.
    - Mostrar opciones de grupo.
8. Más
    - Salir de la App / Logout
    - Búsqueda de usuarios de la App.
    - Muro de ofertas.
    - Información de la cuenta.
9. Más (**Versión Avanzada**)
    - Obtener número de teléfono para el servicio FreePhone.
    - Número gratuito para autentificación de llamadas.
    - Activar versión avanzada.

Los requisitos no funcionales son características del sistema (fiabilidad, mantenibilidad, etc.) que se tienen en cuenta en la fase de diseño.

Se definieron una serie de requisitos no funcionales básicos para el desarrollo de la App. Para la elaboración de los mismos se tuvieron en cuenta conceptos como accesibilidad, conectividad e interoperabilidad.

- La App recibirá notificaciones push del servicio oficial de C2Call.
- La App será únicamente diseñada para dispositivos iPhone, aunque también podrá ser utilizada en dispositivos iPad.
- Se utilizará un entorno de Parse para el almacenamiento de datos.
- Se utilizarán APIs públicas de C2Call para el diseño de la App.
- La App solo responderá, si el dispositivo cuenta con conexión a internet.

Muchas veces, la mejor forma de identificar estas funcionalidades, es analizando las tecnologías existentes y realizando una investigación detallada sobre los posibles servicios ofrecidos en la web, muchos de los cuales, darán soluciones y funcionalidades a nuestra aplicación.
Gracias a esta investigación, encontramos dos servicios que serán esenciales a la hora de desarrollar la aplicación.

3.1. C2CALL

C2Call ofrece a los usuarios una alternativa simple para comunicarse por internet, desarrolla vídeo chat, llamadas de voz (VoIP) y ofrece servicios de mensajería para la computación en la nube desde 2008.
Como uno de los pioneros en la comunicación móvil sobre Protocolo de Internet, lanzó en 2009 la herramienta **FriendCaller**, consiguiendo un reconocimiento inmediato por su innovadora contribución a la industria de las redes sociales.

A diferencia de otros servicios de telefonía basados en VoIP, FriendCaller no tiene que ser instalado, opera en base a un modulo integrado conocido como CallMe-Link. Este enlace establece una conexión instantánea entre los usuarios y se pueden enviar a través de mensajes o correo electrónico. Además, casi todos los navegadores soportan los enlaces CallMe-Link, sin ralentizar la conexión y sin necesidad de establecer cuotas de ningún tipo, consiguiendo que la Red telefónica de FriendCaller comprenda más de 400 países, con territorios repartidos por América del Norte, Europa y Asia.

Después de perfeccionar la plataforma FriendCaller, C2Call empleo una gran cantidad de tiempo y recursos para garantizar la compatibilidad de su red con una amplia gama de dispositivos informáticos y móviles, consiguiendo desarrollar la plataforma C2Call GmbH y su SDK.

Gracias a eso, el SDK de C2Call está disponible, de forma gratuita, para todos los desarrolladores de aplicaciones móviles. Mediante la integración de la tecnología de C2Call en sus aplicaciones móviles, los desarrolladores pueden tener a sus usuarios comunicados a través de múltiples aplicaciones y en múltiples plataformas.
Un usuario de la app puede utilizar el chat, opciones de mensajería multimedia, audio de grupo, llamadas de vídeo y compartir información de ubicación con otros usuarios, todo dentro de una aplicación y gratis.

C2Call también ofrece una solución alternativa para evitar bloqueos UDP y conseguir una comunicación fluida y sin cortes. Actualmente, la mayoría de los operadores de telefonía móvil permiten llamar a sus clientes a través de IP. Sin embargo, muchos usuarios FriendCaller y usuarios de App todavía experimentan bloqueos en los puertos UDP. Algunos gobiernos y operadores no aceptan la idea de que un ciudadano pueda comunicarse libremente y a bajo coste a través de Internet.

Por lo tanto, recientemente C2Call integra una solución alternativa usando TCP, el SDK C2Call ahora soporta un túnel TCP / IP. Esto permitirá a los desarrolladores, que sus aplicaciones sigan funcionando a través de servidores de seguridad donde se bloquea los puertos UDP, garantizando que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

Esta mejora, permite a sus usuarios conectarse donde otras aplicaciones no pueden, a causa de un firewall o un router de restricción forzada.
En el caso de una conexión WiFi, si el UDP está bloqueado por el ISP en el nivel de router o switch, la aplicación se comunica fácilmente a través del túnel TCP.

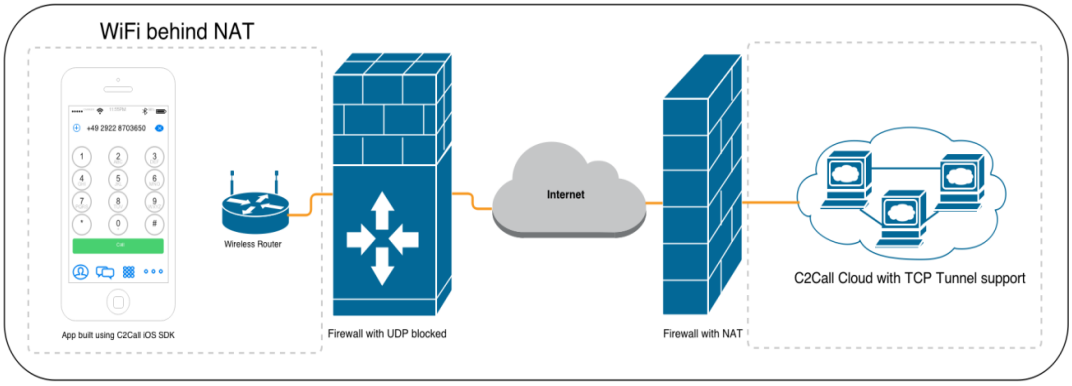


Ilustración 3. Túnel TCP para evitar bloqueo sobre WiFi.

El mismo escenario, esta vez el transportista ha impuesto una restricción al bloquear el puerto UDP (normalmente 5060) para VoIP , la aplicación se conectará de manera normal al servidor C2Call Nube sobre el túnel TCP / IP.

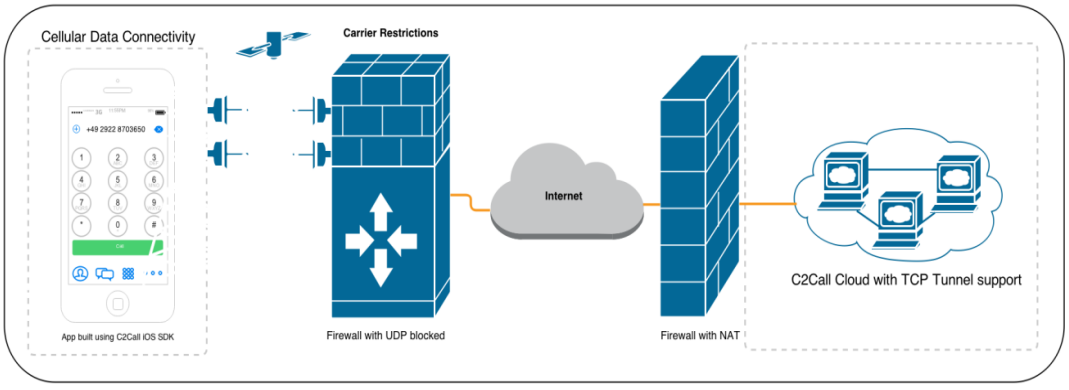


Ilustración 4. Túnel TCP para evitar bloqueo sobre datos.

Todas las aplicaciones que integren iOS SDK C2Call, detectarán automáticamente las situaciones donde se bloquea UDP y establecerán un túnel TCP en su lugar. Toda la comunicación será enviada a través del túnel, incluyendo voz, texto y transferencia de vídeo.

El nuevo SDK v1.2.0 se centra en el apoyo a la nueva iOS 8 y el nuevo iPhone 6, permitiendo las nuevas arquitecturas ARM64 y ARMV7s, con soporte de 64-bits.

Asimismo, los desarrolladores encontrarán 18 nuevas APIs, más relacionados con la comunicación chat, gestión de crédito para llamadas, Login / Register y mucho código fuente de los componentes GUI, que facilitará la personalización de tu app.

Gracias a las maravillas que ofrece C2Call, podremos elegir las funcionalidades necesarias para nuestra aplicación, incluso plantearnos algunas nuevas, logrando diseñar una aplicación sencilla en muy poco tiempo.

3.1.2 VENTAJAS E INCONVENIENTES

Primero empezaremos enumerando las ventajas que ofrece el SDK de C2Call y que nos convencieron para su uso en este proyecto.
Las ventajas más destacables son:

- **C2Call FreePhone**: Llamadas gratis a teléfonos fijos y teléfonos móviles en todo el mundo.
- **Créditos libres**: Que pueden ser ganados por completar las ofertas, ver vídeos y descargar aplicaciones geniales.
- **Servicio de identificación de llamada**: Con el teléfono gratuito de verificación de número, podemos identificar todas las llamadas recibidas.
- Historial de llamadas muy detallado.
- Convierte cualquier iPod touch o iPad en un iPhone, al no requerir una tarjeta SIM.
- **Smart Dialer**: Códigos de área con información de localización automatizada.
- **Muestra tarifas de las llamadas**: Enseña el coste de una llamada tan pronto como el usuario marca el número de teléfono.
- **Servicios de computación en la nube**: Funcionalidades de chat, vídeo llamadas, mensajería instantánea, etc.

Como todos los grandes servicios, C2Call puede ofrecer innumerables ventajas a la hora de diseñar y desarrollar funcionalidades de comunicación, pero tiene muchos inconvenientes si no deseas utilizar los diseños ofrecidos o añadir mejoras al código.
Los inconvenientes más importantes son:

- **Gran cantidad de warnings**: Muchos de ellos originados en las interfaces de usuario y por el uso de métodos obsoletos.
- **Problemas al rediseñar las interfaces**: Todas las APIs ofrecidas para el diseño, tienen establecido un contenedor que guarda la altura, anchura, márgenes, etc. Debido a su estricto diseño, intentar modificar o añadir nuevas características a la interfaz, causará un verdadero dolor de cabeza a cualquier desarrollador principiante que intenté tal osadía.
- **Limitada cantidad de código fuente**: Aunque en la última versión del SDK se han añadido algunos códigos fuentes de los componentes GUI, todavía hay muchas partes que no pueden ser cambiadas.

- **Limitado uso de las notificaciones**: Se podría decir que las notificaciones en C2Call son excepcionales, usan el servicio GCM de Google para que todos los dispositivos reciban las notificaciones, pudiéndote olvidar de su desarrollo. Pero por desgracia, si necesitas enviar notificaciones a otros dispositivos por alguna razón diferente a las propias de su servicio, es simplemente imposible.
- **Nulo control sobre la persistencia de datos**: El mayor defecto a mi parecer, es la nula existencia de funcionalidades que permitan al usuario guardar el historial de chats.
- **Tutoriales con fallos**: Algunos tutoriales indican acciones que pueden llegar a malograr mucho la realización del proyecto, y los cuales, no son detectables para un desarrollador principiante.

3.2 PARSE

Parse es un Back-end, que nos provee de herramientas dentro de un servidor web, para poder implementar en nuestras aplicaciones unas determinadas funcionalidades, es decir, un ecosistema basado en la nube que te permite crear aplicaciones web o nativas altamente escalonadas.

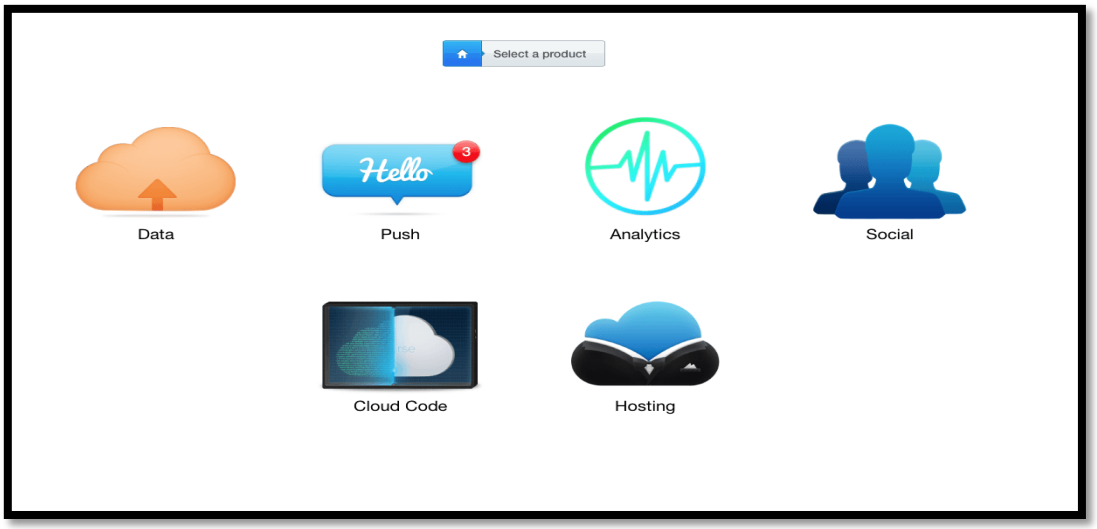


Ilustración 5. Herramientas ofrecidas por Parse

Parse nos provee de las siguientes herramientas:

- Data
- Push
- Analytics
- Social
- Cloud Code
- Hosting

Con ellas podrás analizar el uso que los usuarios le dan a tu App, puedes enviar notificaciones, puedes almacenar archivos o datos y recuperarlos de forma sencilla,

aprovechar las redes sociales, como por ejemplo, que los usuarios de tu App hagan login en tu aplicación por medio de Facebook.

Para poder enviar a uno de nuestros usuarios una notificación, es imprescindible que contemos con lo siguiente:

- Debemos haber registrado en Parse el dispositivo de usuario, dado que necesitamos información identificativa del mismo.
- Debemos haber solicitado autorización, y este debe haber aceptado, recibir notificaciones push de nuestra app.
- Debemos tener un ID de desarrollador de Apple.

Dentro de Parse existen hasta tres formas de enviar notificaciones push a nuestros usuarios (siempre que hayan registrado su dispositivo aceptando la recepción de notificaciones)

- Envío desde el panel de Parse.
- Envío mediante el API provista por Parse.
- Envío mediante el uso de Clound Code.

Todas fáciles de implementar, basta con dirigirnos a la pestaña de **Push Notifications** y pulsar el botón **Send a push** de la parte superior derecha, para desplegar una pantalla donde podemos escoger todo lo necesario.

Parse también ofrece muchas formas de usar su servicio. Puedes usar un conjunto de APIs que están escritas como librerías nativas. Aquellos que desean crear sus app usando tecnologías nativas, pueden tomar ventaja de los SDKs completamente especializados para su plataforma de desarrollo. (Por el momento hay SDKs disponibles para Android, iOS y Windows).

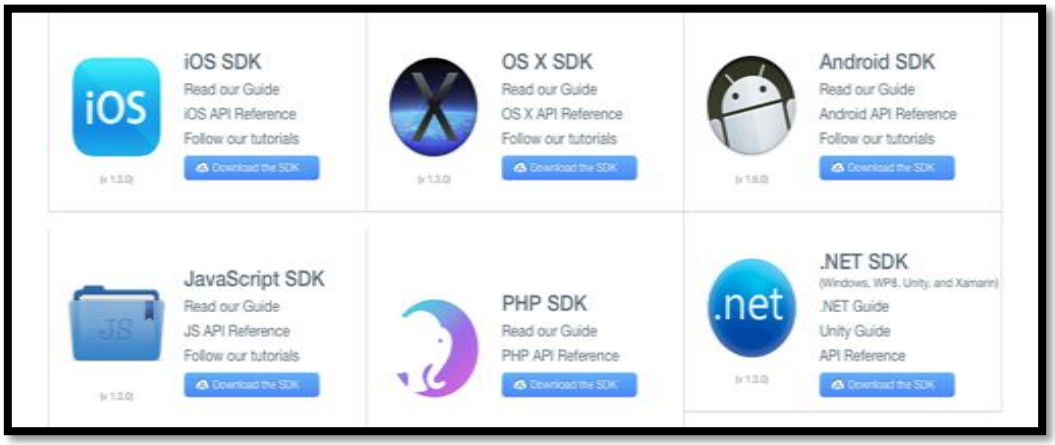


Ilustración 6. Documentación de Parse.

Uno de los beneficios de Parse es la posibilidad de omitir por completo el trabajo con un servidor, aunque aún puedes usar un servidor web tradicional. Con el API de REST, tu servidor puede comunicarse con Parse para buscar y trabajar con datos.

3.2.1 VENTAJAS E INCONVENIENTES

Primero empezaremos explicando las ventajas que ofrece Parse respecto a otros Back-end. Las ventajas más destacables son:

- **Excelente Documentación:** Una de las cosas a destacar de Parse es la excelente documentación que tienen en su web, ordenada por dispositivos (iOS, Android, servidor, etc.) y con muchos ejemplos y aplicaciones de prueba.
- **Permanencia de datos:** Parse permite que los usuarios almacenen datos en sus servidores. Los datos pueden ser de cualquier tipo.
- **Notificaciones:** Este apartado es para quitarse el sombrero, pues de otra manera, sin usar Parse, el envío de notificaciones a los usuarios de nuestras Apps se convierte en una tortura. Apple controla mucho la seguridad, es decir que para enviar notificaciones desde tu propio servidor tienes que tener autenticación segura, establecer un protocolo cifrado con los servidores de Apple, y no sé cuantas cosas más. Con Parse todo es mucho más sencillo, sólo debes subir a su web unos certificados que previamente has de crear en iOS Dev Center y listo.
- **Estadísticas:** En cuanto al tema de estadísticas, reconozco que en este tema todavía están en ello, hay otros servicios web mejores para analizar el uso que un usuario hace de tu App, pero si ya trabajas con Parse, no está de más incluir esta funcionalidad en tu App, para tener todo condensado en un mismo Back-end.
- **Ahorro de tiempo:** Como he dicho antes, podemos hacer lo mismo sin Parse, pero deberíamos contratar un servidor, y luego montar toda la infraestructura para los servicios, con Parse nos ahorramos todo eso.

A diferencia de C2Call, no he encontrado inconvenientes en usar Parse, todos los servicios que ofrece son muy ventajosos, quizás solo una pega, que no tenga un servicio de comunicación de chat.

4. DISEÑO E IMPLEMENTACIÓN

El diseño es la primera de las tres actividades técnicas que implica el desarrollo de cualquier aplicación o sistema de ingeniería, estas etapas son diseño, Implementación y pruebas. El objetivo del diseño es producir un modelo o representación de una entidad que se construirá posteriormente. Hay tres características que sirven como parámetros generales para la evaluación de un diseño. Estos parámetros son los siguientes:

1. El diseño debe implementar todos los requisitos explícitos obtenidos en la etapa de análisis.
2. El diseño debe ser una guía que puedan leer y entender los que construyen el código y los que prueban y mantienen el software.
3. El diseño debe proporcionar una idea completa de lo que es el software.

Para cumplir estos parámetros dividiremos el proyecto en cuatro etapas:

1. La interfaz principal, desarrollada con las APIs de C2Call.
2. La interfaz secundaria, diseñada con los métodos ofrecidos por el SDK de iOS y que permitirá controlar las funciones del foro.
3. Diseño de la base de datos y de los métodos necesarios para la interfaz secundaria, desarrollado con los servicios de Parse.
4. Compatibilización de diseños e introducción de segundo idioma.

Durante estos cuatro puntos, se explicará detalladamente todos los pasos seguidos para su diseño e implementación, teniendo en cuenta que en algunas fases, ambas etapas serán simultáneas.

4.1 INTERFAZ PRINCIPAL

Para diseñar la interfaz principal de nuestra app, vamos a usar las APIs que ofrece C2Call en su SDK. Una API (siglas de 'Application Programming Interface') es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.
Las API pueden servir para comunicarse con el sistema operativo, con bases de datos o con protocolos de comunicaciones.

Las API son valiosas, ante todo, porque permiten hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar reinventando la rueda constantemente, reutilizando así código que se sabe que está probado y que funciona correctamente.

4.1.1 FASE 1: REGISTRO EN C2CALL

Primero empezaremos registrándonos en la página de C2Call para poder descargar su SDK y registrar el nombre de nuestra aplicación.

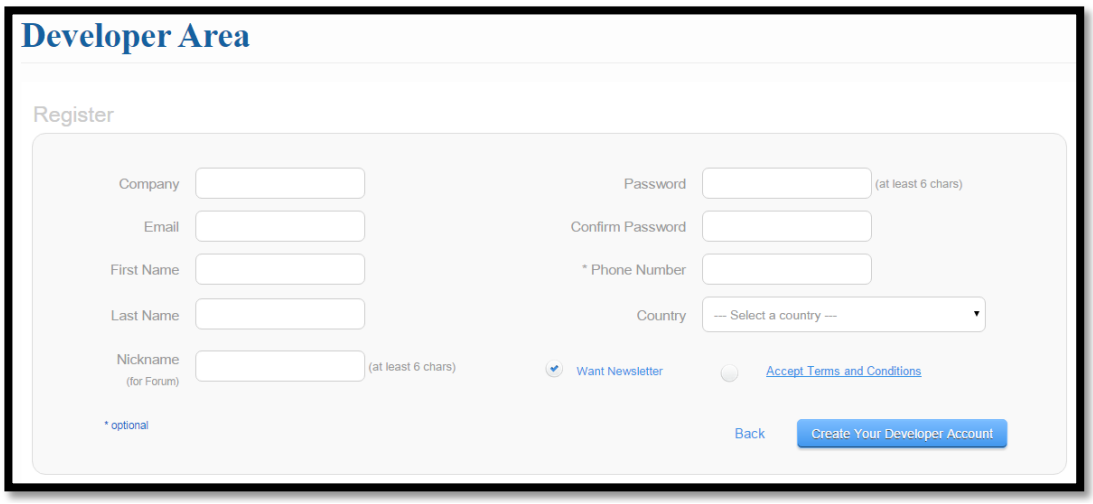


Ilustración 7. Pagina de registro de C2Call.

Una vez terminado el registro, entramemos en la pantalla de bienvenida, donde podremos descargar el SDK de C2Call y crear nuestra aplicación.

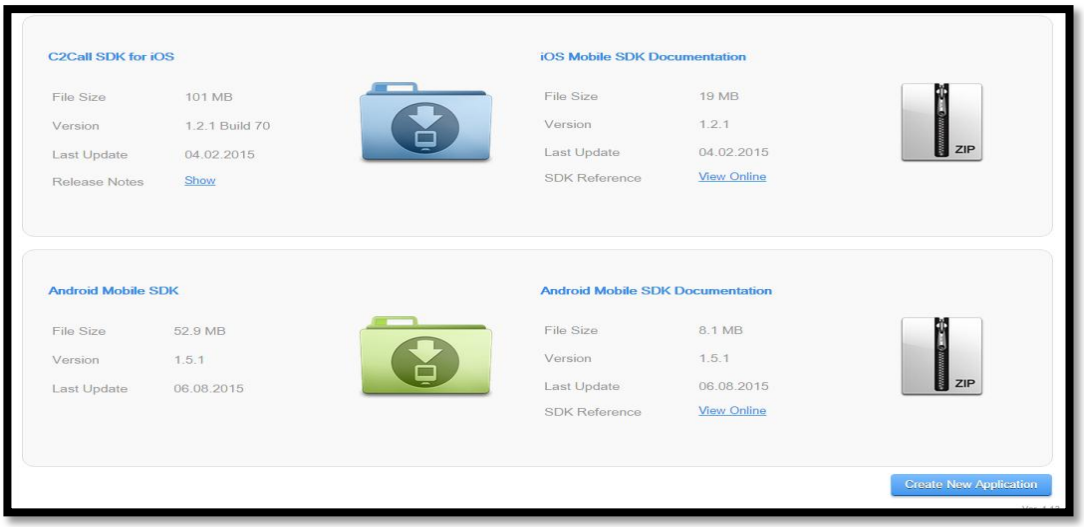


Ilustración 8. Zona de descarga del C2Call SDK.

Antes de continuar con el registro de la aplicación, tendremos que descargar el SDK y crear nuestro proyecto en xcode, porque más adelante nos pedirán algunos datos relacionados con el.

El primer paso será copiar la carpeta descargada de C2Call en nuestro escritorio, para tenerla a mano. Después abrimos el xcode y pulsamos el botón para crear un nuevo proyecto.

Ilustración 9. Botón para crear un nuevo proyecto.

Elegimos **Tabbed Application** en la nueva pantalla y pulsamos **Next**.

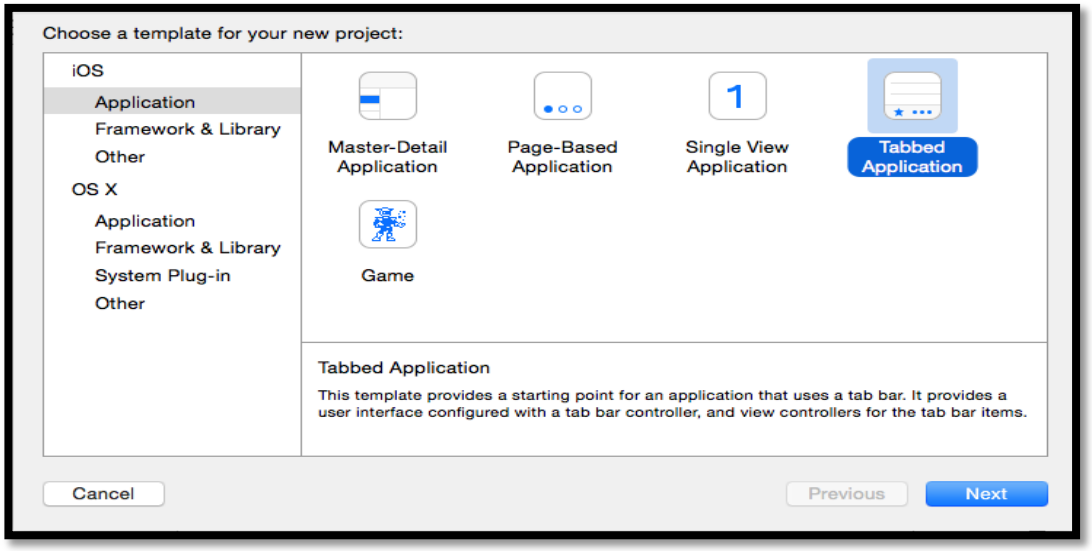


Ilustración 10. Crear nuevo proyecto en Xcode.

Introducimos el nombre que le queremos dar a nuestro proyecto, el nombre de la organización y un identificador de la compañía (el nombre introducido aparecerá debajo en **bundle identifier**, que servirá como un identificador único de nuestra app).

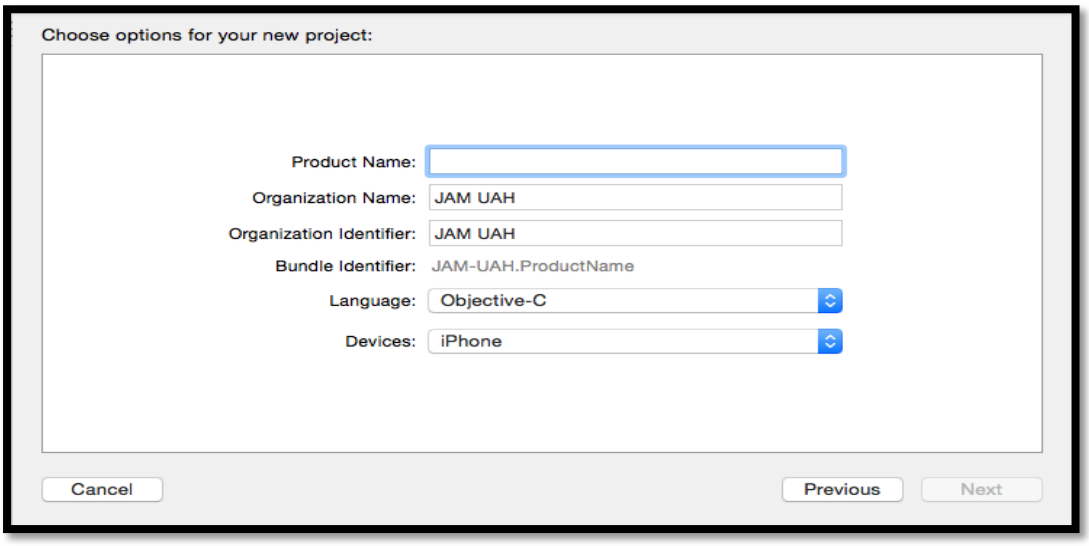


Ilustración 11. Elegir nombre de proyecto en Xcode.

En esta pantalla tendremos la opción de elegir el lenguaje utilizado para desarrollar la aplicación y el tipo de dispositivo al que estará destinado.

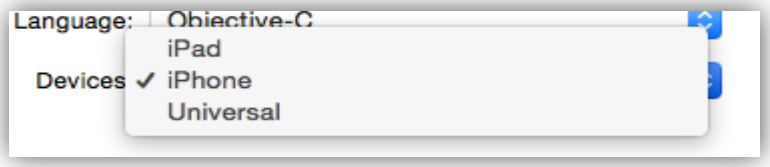


Ilustración 12. Opciones de Devices.

Si elegimos la opción de **Universal**, la app podrá funcionar en los dos dispositivos, pero debido a las diferencias evidentes de tamaño entre un iPhone y un iPad, muchas guías de desarrolladores recomiendan diseñar la app exclusivamente para cada plataforma, de esta forma se podrá usar las herramientas específicas para iPad que mejoran la experiencia de usuario y la usabilidad.
Siguiendo este consejo, he decidido diseñar la app exclusivamente para iPhone.

Una vez rellenado los datos, pulsamos **Next** y nos preguntará donde queremos guardar el proyecto. Elegimos el destino y le damos al botón de **Create** para crear nuestro proyecto.

Ahora que ya tenemos creado el proyecto, seguiremos con el registro de C2Call. Al pulsar **Create New Application**, pasaremos a otra pantalla para crear la app, donde tendremos que introducir el nombre de nuestro proyecto, una pequeña descripción sobre la app que vamos a desarrollar, subir el identificador de nuestro proyecto y elegir la plataforma donde implementaremos nuestra aplicación.

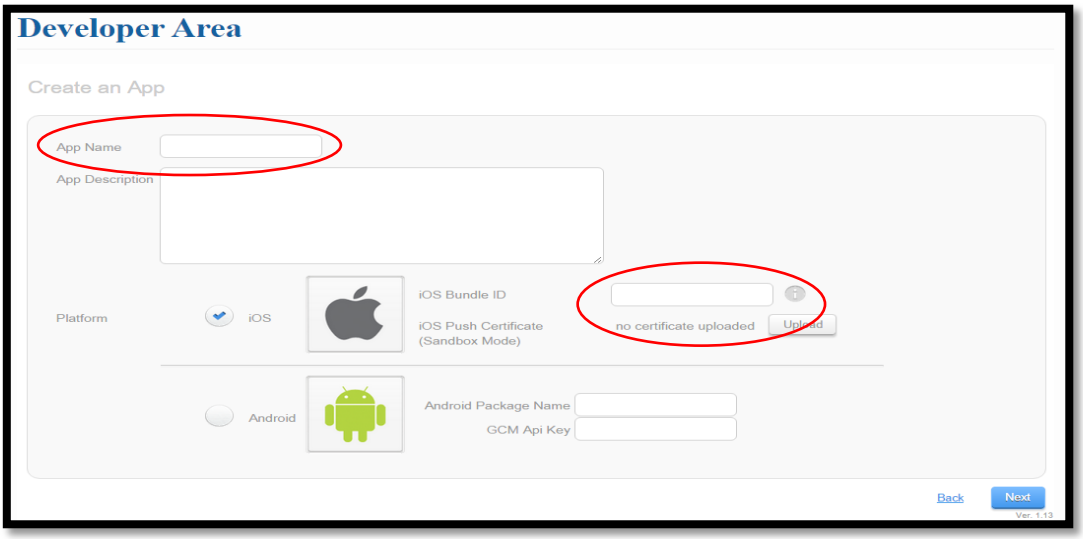


Ilustración 13. Crear una App en la página de C2Call.

Aquí hay dos partes muy importantes a destacar:

- El nombre que pongamos tiene que coincidir con el de nuestro proyecto creado en xcode.
- Tenemos que introducir el **Bundle Identifier** que se encuentra en la vista de resumen de nuestro proyecto xcode.

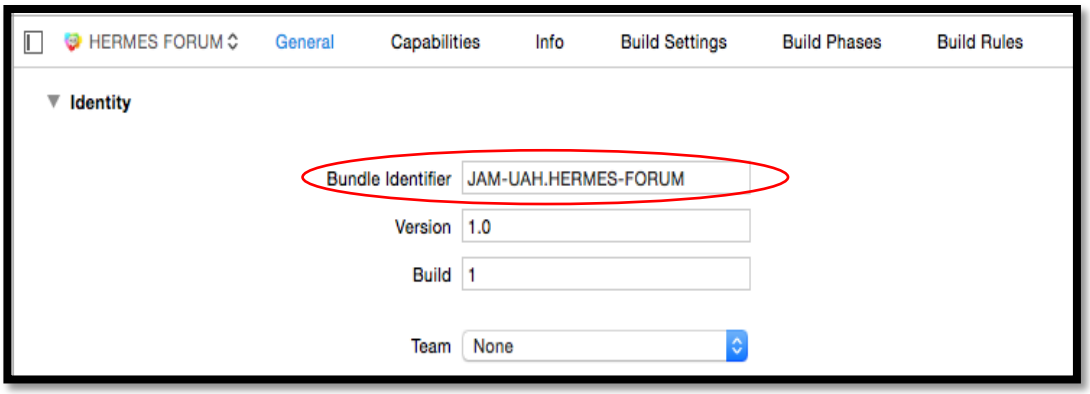


Ilustración 14. Bundle identifier en Xcode.

Una vez rellenado, pulsamos **Next** y nos remitirá a otra pantalla donde tendremos que elegir los servicios que deseados implementar en nuestra aplicación.

Llamadas VoIP, mensajería instantánea, vídeo llamadas, etc.

Ilustración 15. Opciones de C2Call.

Nosotros seleccionaremos todas las opciones menos conexión Facebook y publicidad InApp, para las que son necesarios tener una cuenta Pro. Más adelante podremos cambiar las opciones seleccionadas si deseamos.

Le damos a **Next** y pasaremos a la última etapa, donde te da la opción de elegir la empresa que se encargará de controlar los anuncios InApp, como nosotros no tenemos la opción, le damos al botón de **Create Aplication** y listo.

Ahora que ya tenemos creada la aplicación, nos saldrá la pantalla principal de nuestra cuenta, donde podremos ver nuestro perfil, las opciones de publicidad InApp y un rótulo de Apple con el nombre, identificador, estado y número de usuarios de nuestra aplicación.

Si apretamos el rótulo, nos mostrar las opciones principales de la aplicación. Donde podremos descargar el C2Call-SDK, cambiar las opciones de la app y ver la información de los usuarios registrados en nuestra aplicación. Además, podremos editar o borrar

los datos de los usuarios, pero no crear un usuario directamente desde la pagina de C2Call.

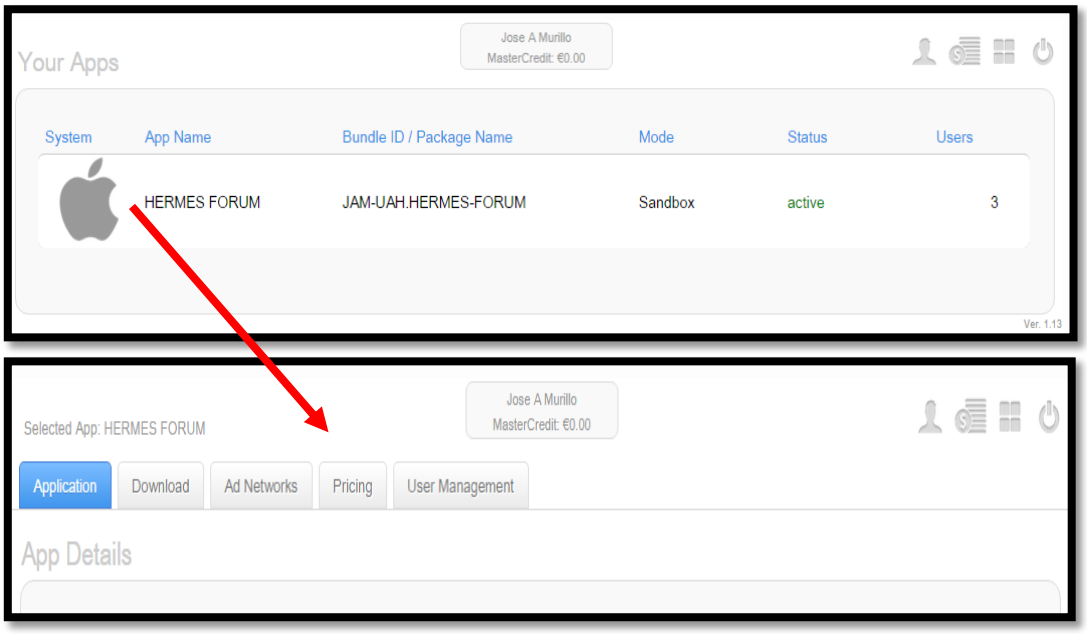


Ilustración 16. Pantalla principal de C2Call.

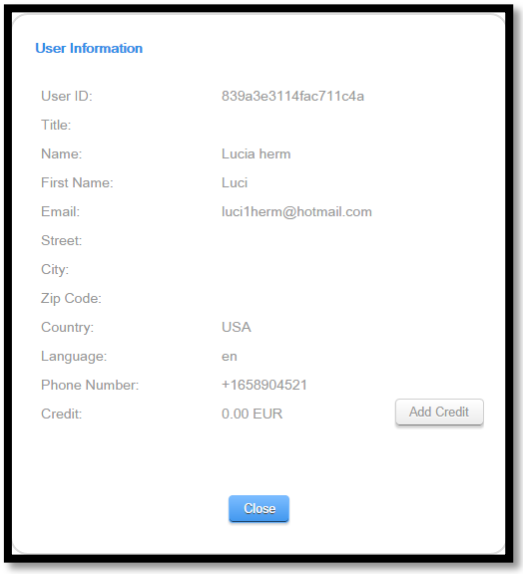


Ilustración 17. Información de usuario en C2Call.

Ahora que ya tenemos registrada nuestra aplicación, solo necesitamos diseñar la interfaz principal en nuestro proyecto de xcode, con la ayuda de los controladores predefinidos del C2Call SDK.

4.1.2 FASE 2: DISEÑO PRELIMINAR

Para poder usar las características de C2Call, primero tendremos que añadir a nuestro proyecto de xcode los archivos necesarios.

¿Pero cuáles son los archivos necesarios?

Gracias a dios, C2Call incluye en su SDK un conjunto de tutoriales y ejemplos que podemos usar para empezar a diseñar nuestra interfaz.

Empezaremos siguiendo los pasos indicados en el tutorial Simple-Chat-App. Que nos explica como diseñar una pequeña aplicación con Login / Register, una agenda de amigos, la funcionalidad de chat, opciones de menú y desconexión. Como todo lo indicado son requisitos que buscamos para nuestra aplicación, es como matar dos pájaros de un tiro.

Primero buscaremos el ejemplo deseado. Abrimos la carpeta **C2Call-SDK-V1.2.1-B70** → **SampleCode** → **SDK-SimpleDialer Sample** → **SDK-SimpleDialer Sample.xcodeproj**

Seleccionamos todos los frameworks dentro de la carpeta Frameworks, perteneciente al ejemplo SDK-SimpleDialer, arrastramos y la soltamos en nuestro proyecto. De esta forma tenemos todas las bibliotecas necesarias para ejecutar nuestro proyecto.

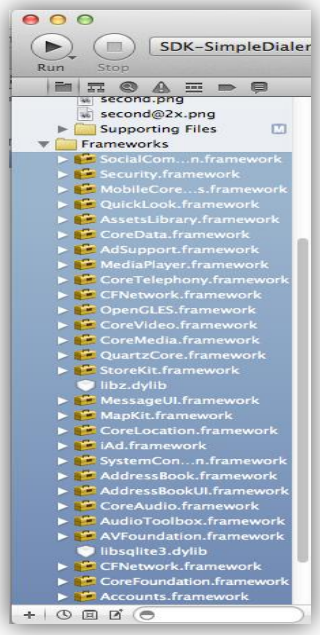


Ilustración 18. Librerías C2Call.

Al soltar los frameworks en nuestro proyecto, saldrá una pantalla indicado si deseamos crear una referencia para los nuevos archivos. Dejamos los valores señalados por defecto y marcamos crear copia en destino, de esta forma los archivos estarán dentro

de nuestro proyecto y no necesitaremos la carpeta externa. Así conseguiremos una mayor portabilidad, por si es necesario instalar el proyecto en otro ordenador.

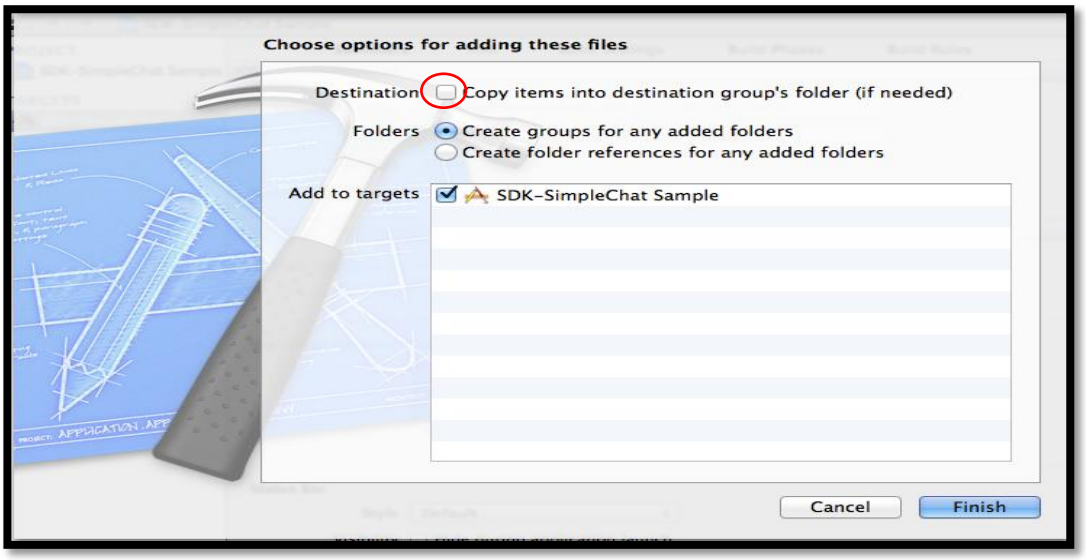


Ilustración 19. Pantalla de copia de archivos en xcode.

Siguiendo el mismo proceso, arrastraremos la carpeta que está dentro de C2Call-SDK, llamada **SocialCommunication.ressources**, a nuestro proyecto de xcode y la soltaremos dentro de nuestra carpeta **Supporting Files**. Saldrá una pantalla igual a la anterior y volveremos a seleccionar copiar dentro de nuestro proyecto. Esta carpeta contiene todos los iconos y aplicaciones necesarias para las APIs ofrecidas por el SDK.

Después, el tutorial nos indicará cómo realizar unas modificaciones en nuestro **Build settings**, para añadir una ruta de búsqueda de cabecera ("/usr/include/libxml2"), unos nuevos enlaces hacia la ruta (-lxml2 y -lstdc++), cambiar la arquitectura usada en la app y desconectar en nuestro **Main.storyboard** el uso de autolayout.

Los cambios parecen inofensivos, pero dos de ellos causarán innumerables problemas a la hora de desarrollar nuestro proyecto. Estos fallos los comentaremos cuando salgan en la implementación.

El siguiente paso que nos enseña el tutorial, es buscar el **SCStoryboard.storyboardc**, que se encuentra dentro de la carpeta **SocialCommunication.ressources**, añadida con anterioridad en nuestro proyecto. En este archivo se encuentran todas las APIs ofrecidas por C2Call, con las cuales podremos diseñar nuestra aplicación.

Los siguientes pasos en el tutorial, son copiar las APIs necesarias para nuestro ejemplo y pegarlas en nuestro **Main.storyboard**. Las APIs que seleccionaremos son:

- Controlador de lista de amigo (Agenda).
- Encontrar amigos.
- Perfil.
- Offerwall.
- Controlador del chat y todas sus funciones.

Nuestro **Main.storyboard** quedaría igual que la siguiente imagen.

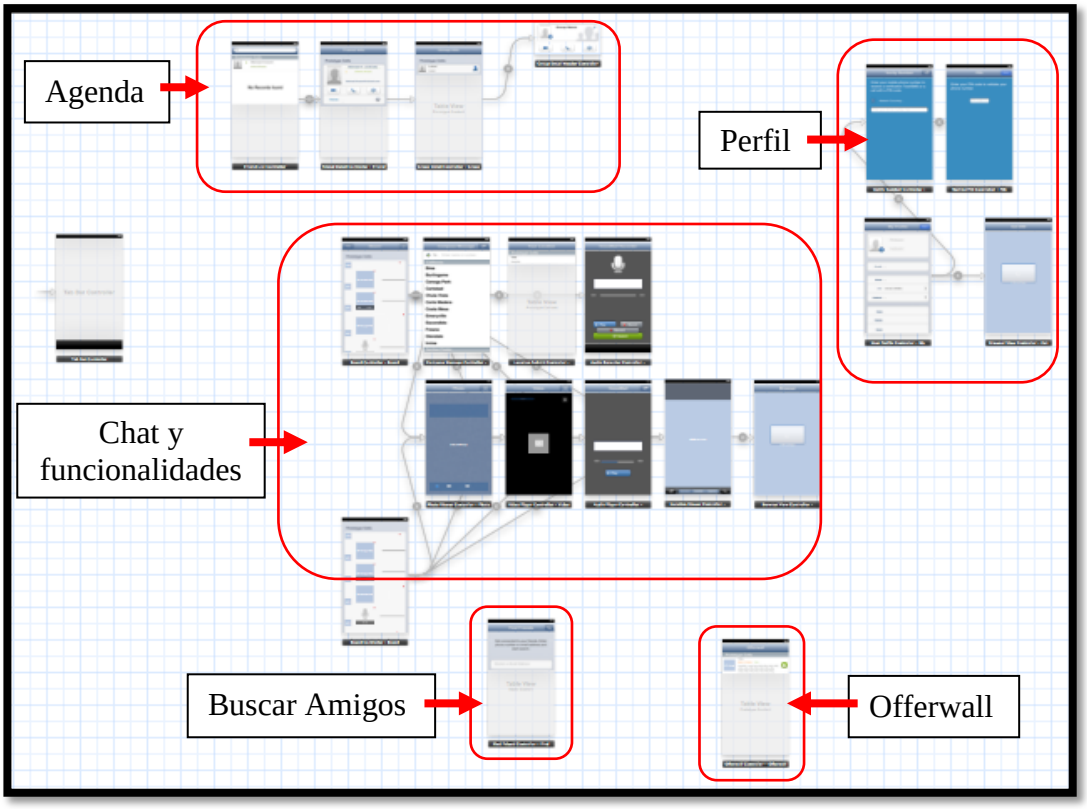


Ilustración 20. MainStoryboard con varias APIs.

Si nos fijamos, hay una interfaz que no ha sido señalada, es la que viene por defecto al crear nuestro proyecto en xcode. Esta es un controlador **Tab Bar**, que puede controlar diferentes pestañas con un menú colocado en la parte inferior del controlador. Lo que vendría a ser nuestro menú principal.

Ahora nos toca unir todas las APIs con nuestro **Tab Bar controller**, primero añadiremos a nuestro **Main.storyboard** varios **Navigation controller**, que servirán de enlace para unir las interfaces. El proceso será el siguiente:

- Seleccionamos un Navigation y apretamos el botón derecho del ratón. Nos saldrá una ventana emergente con el **Triggered Segues** de Navigation.

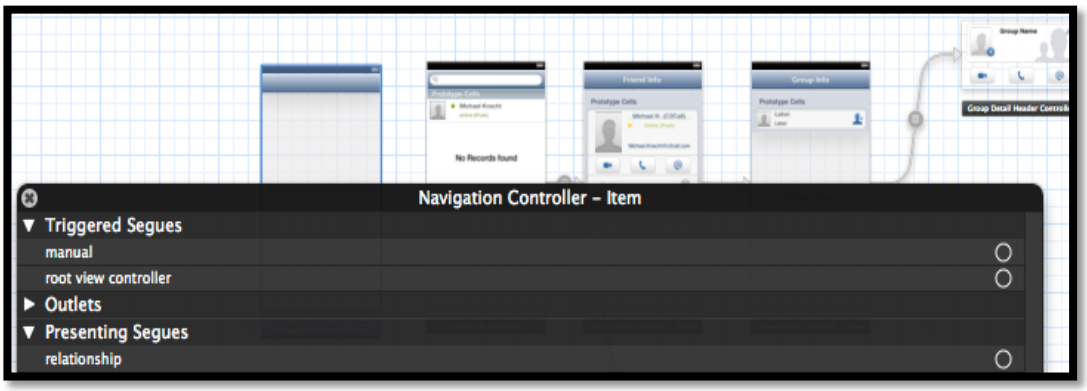


Ilustración 21. Triggered Segues Navigation.

- Seleccionamos **root view controller** con el puntero del ratón en el círculo y lo arrastramos hasta el **view controller** deseado, al soltarlo nos creará un enlace.

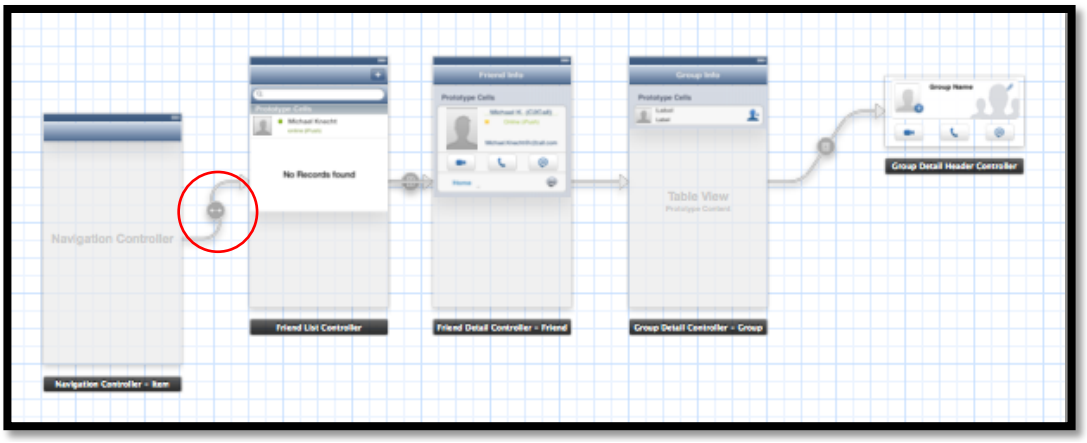


Ilustración 22. Conexión entre dos interfaces.

- Después seleccionamos el **Tab Bar** y apretamos el botón derecho del ratón. Nos saldrá el **Triggered Segues** de Tab Bar.

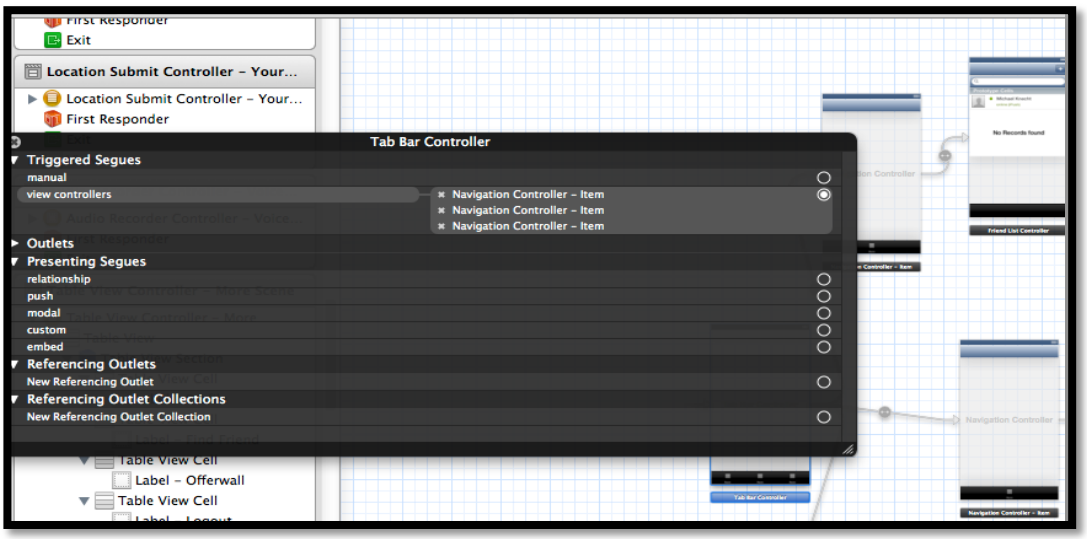


Ilustración 23. Triggered Segues Tab Bar.

- Seleccionamos view controller con el puntero del ratón en el círculo y lo arrastramos hasta el navigation deseado, al soltarlo nos creará un enlace.
- Repetimos este proceso con el chat.

Antes de empezar con la siguiente parte, vamos a buscar las dos últimas APIs que vamos a usar en nuestro primer diseño, Logout y Login/Register, y pegarlas en nuestro Main.storyboard.

Para las APIs de Buscar amigos, Offerwall, Perfil y Logout vamos a utilizar además una **Table View** controller, de esta forma podremos enlazar varias APIs con la tabla y este con nuestro **Tab Bar**, ahorrando espacio al menú principal.
El primer paso será configurar nuestro **Table View** para que tenga 4 filas, para ello seleccionaremos la tabla y en sus opciones de configuración elegiremos **Static Cells**.

Una vez seleccionado, saldrá una **Table View Section** en los componentes de nuestra tabla, la seleccionamos y tendremos la opción de indicarle el número de filas deseadas.

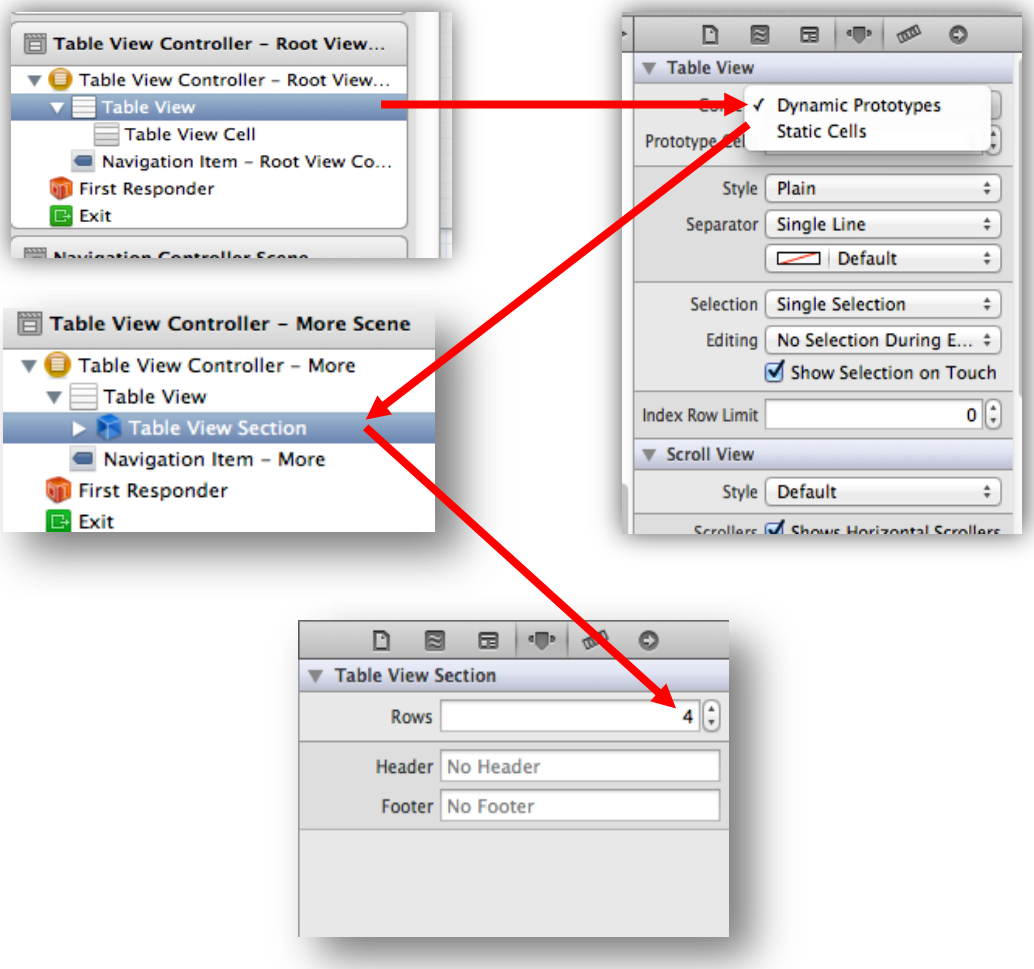


Ilustración 24. Configurar una Table View.

El último paso será cambiar el título de las filas por los nombres de las APIs que vamos a introducir (Logout, Profile, Offerwall y Find Friend) y enlazarlas con sus respectivos controladores.

Para enlazarlas repetimos los pasos anteriores, seleccionamos una celda de la tabla y pulsamos el botón derecho del ratón. Nos saldrá el **Triggered Segues** de la celda. Elegimos la opción de **selection** con el puntero del ratón en el círculo y lo arrastramos hasta el controlador deseado. Repetimos este proceso cuatro veces, para cada uno de los controladores.

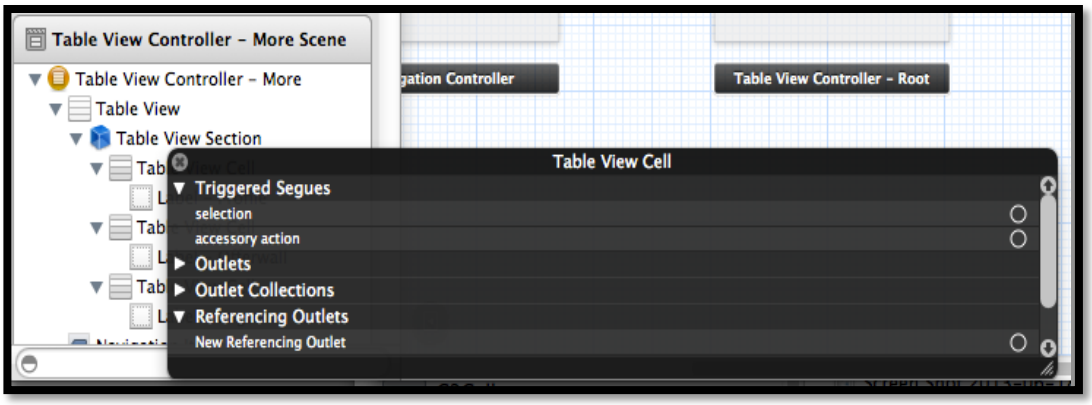


Ilustración 25. Triggered Segues Table View.

Ya solo nos falta unir el Login/Register con nuestro **Tab Bar** y tendremos todas las APIs conectadas y listas. Pero para este último enlace realizaremos unos pequeños cambios.

Este último controlador tiene una rasgo especial, si el usuario ya está registrado o conectado en la app, no se mostrará la pantalla de inicio.

Para que la API pueda realizar la comprobación, es necesario añadirle un identificador al enlace creado. Para establecer el enlace, seguimos los mismos pasos usados para la Agenda y el Chat, después seleccionamos el **Segue** y le añadimos un identificador.

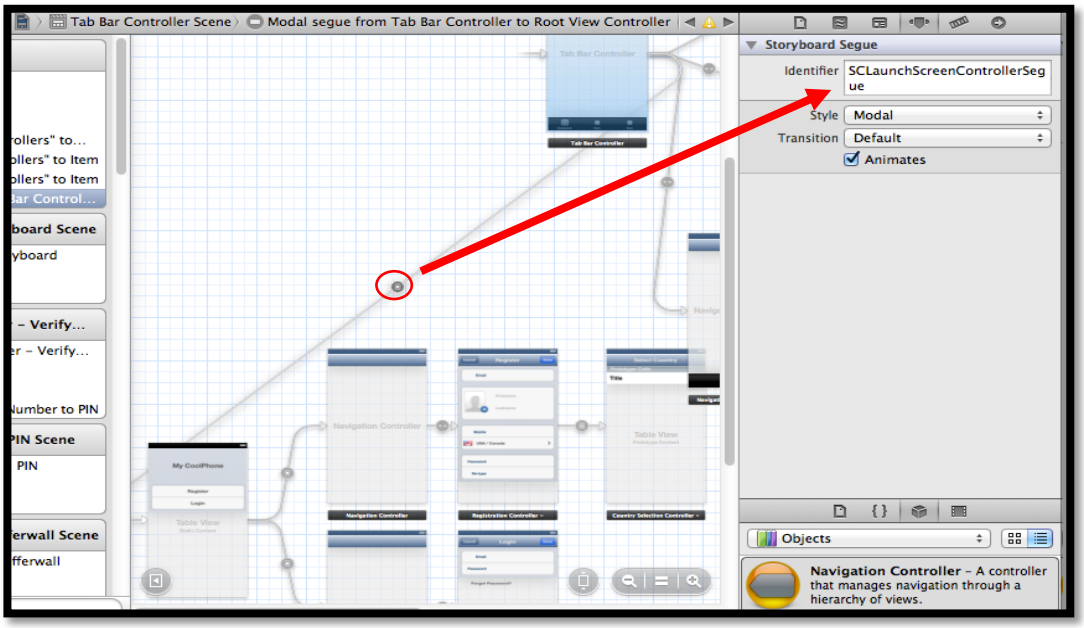


Ilustración 26. Añadir un identificador al Segue.

El identificador, es esencial para poder distinguir todos los enlaces. Si varias interfaces de usuario están interconectadas, la app tiene que saber la información que envía a cada una.

Con este último paso terminamos de crear el diseño preliminar de la aplicación, pero aún nos falta introducir los cambios necesarios en el código y realizar algunas pequeñas pruebas de viabilidad.

4.1.3 FASE 3: CÓDIGO Y PRUEBAS

Una vez terminado el diseño, el tutorial nos dice que tenemos que copiar dos archivos a nuestro proyecto, **SPLogoutViewController.h** y **SPLogoutViewController.m**, que se encuentran en la carpeta principal del ejemplo. También nos comenta, que tenemos que modificar los dos archivos establecidos por xcode al crear un proyecto.

Los archivos creados por defecto en xcode son **AppDelegate.h** y **AppDelegate.m**, el tutorial nos facilita unas líneas de código que tenemos que añadir:
En nuestro .h:

- Importamos una cabecera de las librerías de C2Call.
  #import <SocialCommunication/SocialCommunication.h>
- Le decimos a nuestra clase **AppDelegate**, que se ajuste al protocolo indicado.
  @interface AppDelegate : C2CallAppDelegate

En nuestro .m implementaremos una serie de funciones que conectaran nuestra aplicación con los servicios de C2Call, comprobaran si hay conexión, etc. La más importante es:

- Esta función envía nuestro número de identificación y número secreto para conectarse con los servicios de C2Call, si estos valores no coinciden con los registrados en la aplicación, mostrara un mensaje y se terminara.

```
(BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
        self.affiliateid = @"1F3E9213F51427D53";
        self.secret = @"1bc8c212de4251d53e5f1a414dbc3151";
        return [super application:application
        didFinishLaunchingWithOptions:launchOptions];}
```

Estos valores son facilitados por C2Call al crear una aplicación en su página web.

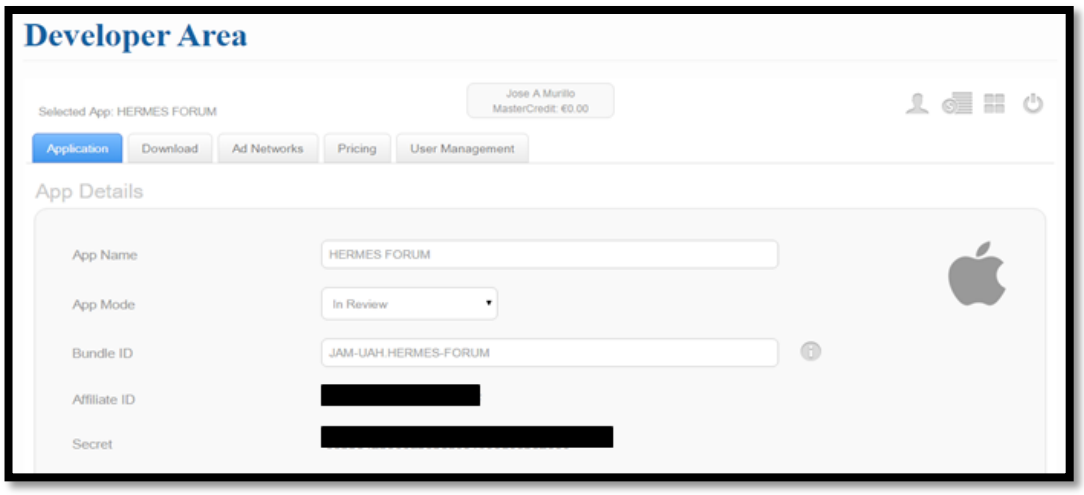


Ilustración 27. Número secreto en C2Call.

El último paso del tutorial es ejecutar la aplicación y comprobar sus funcionalidades. Para ello nos ofrece los datos de dos usuarios de prueba ya registrados, solo es necesario introducir su correo electrónico y contraseña en la pantalla de **Login** y entraremos en la aplicación con todas las funcionalidades activas.

He de resaltar, que la primera vez que ejecute la aplicación salieron 326 warnings y 1 error. Dicho error era producido por unos cambios que realizamos al principio del tutorial, en concreto eliminar de **Architectures** la arquitectura arm64.

En el nuevo xcode, la arm64 se ha convertido en una de las arquitecturas estándar para construir aplicaciones, pero debido a que el tutorial usaba de ejemplo el xcode 4.6.1, te recomendaba quitarla al no ser admitida en su momento por muchas funcionalidades.

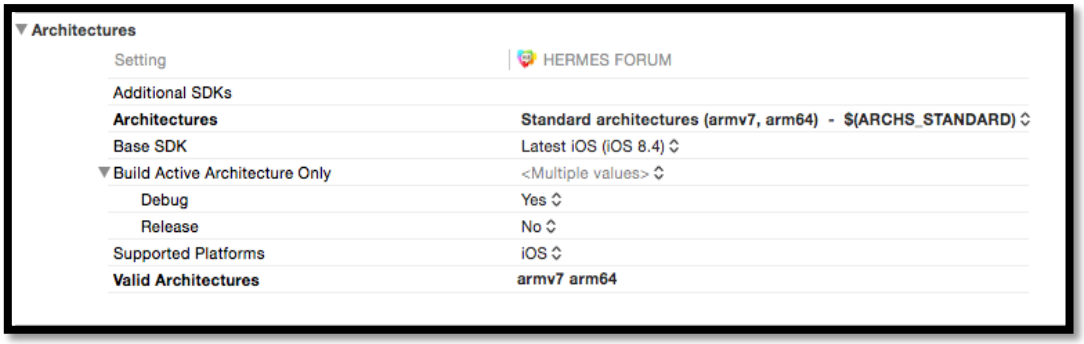


Ilustración 28. Arquitecturas de la aplicación.

Fue necesario emplear 6 horas y varios intentos hasta conseguir solucionar un fallo, que se podía haber evitado renovando el tutorial.
Una vez eliminado el error, la aplicación parecía arrancar bien, pero se terminaba bloqueando al no encontrar algunas librerías o marcos usados en el diseño de las APIs.

Se necesitó emplear 2 horas más para conseguir solucionar los errores relacionados con los marcos y las librerías. La solución implementada fue la siguiente:

Si se trata de un error " **warning: directory not found for option '- L /...** ". Eso significa que es un error de Biblioteca, y los pasos para solucionarlo son:

- Hacer clic en el proyecto (**targets**).
- Hacer clic en **Build Settings**.
- Buscar **Library Search Paths** y eliminar los caminos que ya no son usados.

Si se trata de un error " **warning: directory not found for option '- F/...** ". Eso significa que es un error de Marco, y los pasos para solucionarlo son:

- Hacer clic en el proyecto (**targets**).
- Hacer clic en **Build Settings**.
- Buscar **Frameworks Search Paths** y eliminar los caminos que ya no son usados.

Después de solucionar los últimos errores, la aplicación se ejecutó sin problemas y conseguí examinar todas las funcionalidades correctamente.

Una vez seguro de que la aplicación funcionaba, intenté solucionar todos los warnings. Muchos de ellos eran causados por el diseño de las APIs, quizá debido a que el xcode 6.4 era mucho más concienzudo que su predecesor.

Conseguí eliminar todos los warnings relacionados con la aplicación directamente, es decir, aquellos que salen antes de compilar todos los archivos. Los demás warnings no podían ser solucionados, ya que se debían a los iconos y APIs de C2Call que eran usados de ejemplo en el archivo **SCStoryboard**.

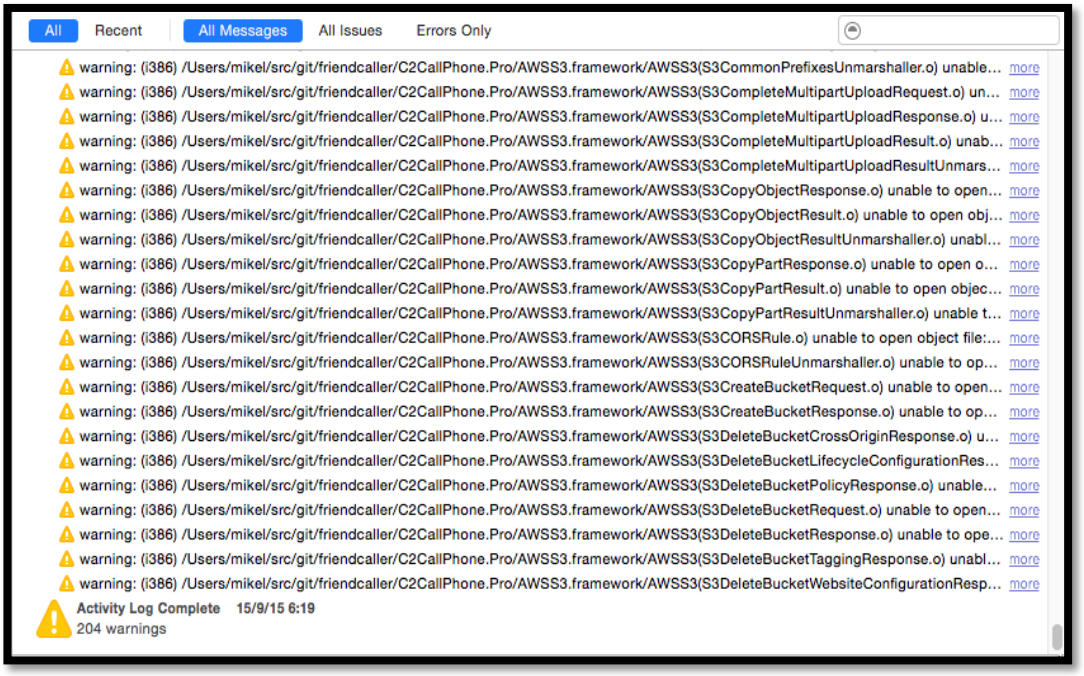


Ilustración 29. Errores producidos por el SCStoryboard de C2Call.

4.1.4 FASE 4: DISEÑO FINAL

Ahora que ya funciona el diseño preliminar, indagaremos entre las APIs ofrecidas por C2Call-SDK, en busca de alguna que cumpla los requisitos establecidos para nuestra aplicación.

Después de comprobar muchos tutoriales y ejemplos, encontramos 3 APIs que nos ayudarán a terminar de cumplir los requisitos deseados. Estos son:

- Forgot Password, para recuperar la contraseña.
- Crear grupos de Chats.
- Añadir fotos al perfil, que en este caso se usará para el de grupos.

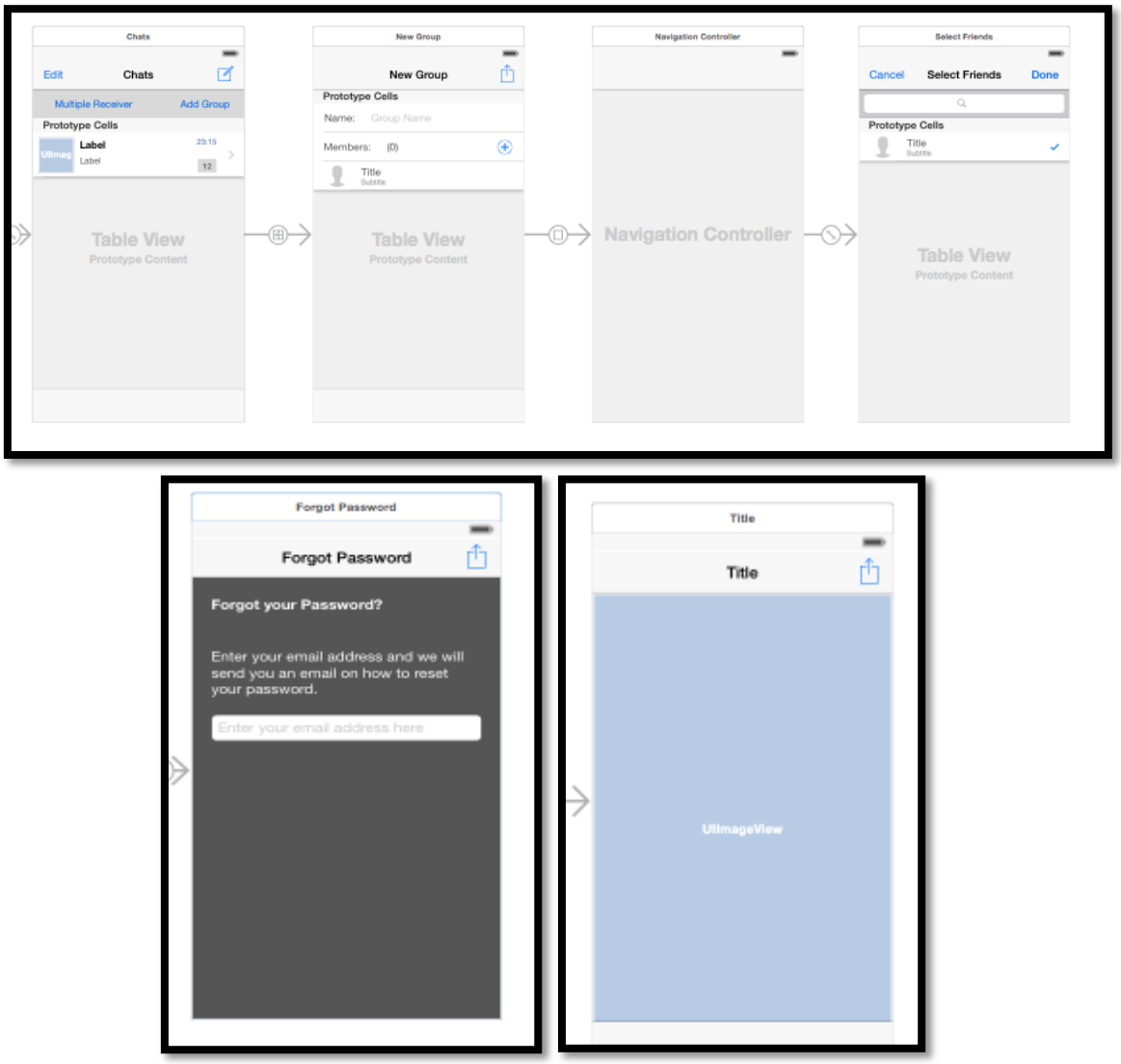


Ilustración 30. Ultimas APIs añadidas al diseño.

Las APIs que nos permiten crear grupos de chat, las uniremos directamente a nuestro Tab Bar. De esta forma son accesibles desde el menú principal.

Para enlazarlas seguiremos los procesos ya conocidos, añadiremos a nuestro **Main** un **Navigation Controller** que servirá de enlace, seleccionamos el navigation y apretamos el botón derecho para mostrar el **Triggered Segues**, etc.

Para enlazar el API de recuperar contraseña, tenemos que ir al interfaz de **Login**, y seleccionar el botón de **Forgot Password?**, apretamos el botón derecho del ratón para mostrar el **Triggered Segues**, seleccionamos la opción de **action** pulsando el círculo y lo arrastramos hasta el **view controller** de recuperar contraseña.

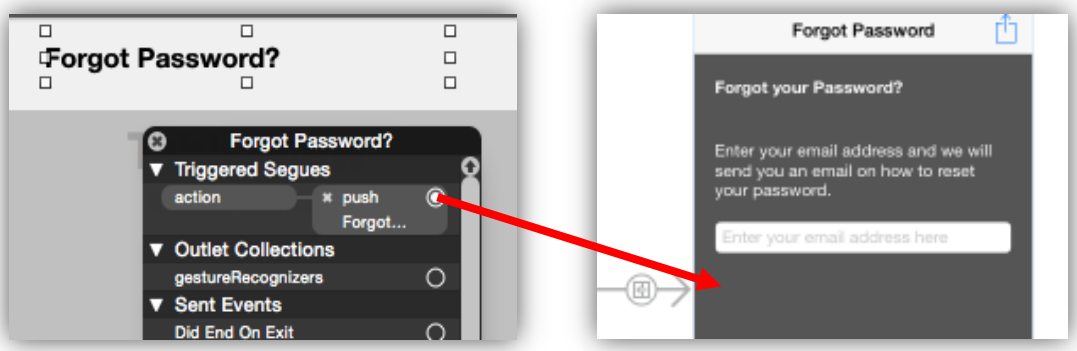


Ilustración 31. Enlazar un botón con su controlador.

Por último enlazaremos el **view controller** de añadir foto, con el último **view controller** del grupo que controla la agenda, de esta forma, podremos añadir una foto a los grupos de chat ya creados.

Con estas funcionalidades y las ofrecidas por las diferentes APIs del ejemplo anterior, conseguimos crear una interfaz principal bastante más completa de la que habíamos planteado en un momento

4.2 INTERFAZ SECUNDARIA

En esta segunda etapa del proyecto, nos dedicaremos a diseñar la interfaz secundaria, con la que podremos cumplir los últimos requisitos establecidos para la aplicación.
Estos requisitos estarán relacionados con las funcionalidades del foro:

- Poder ver los temas existentes en el foro.
- Poder crear un tema nuevo en el foro.
- Poder añadir un comentario en el foro.
- Poder invitar a un desconocido a nuestra lista de amigos.

Para la realización de estas interfaces, usaremos todos los elementos proporcionados por el xcode, pero nuestra prioridad será la funcionalidad y sencillez, para no complicar su afinidad con los diferentes iPhone.
Más adelante explicaremos los pasos realizados para compatibilizar todas las APIs con cada tipo de iPhone.

4.2.1 DISEÑO

En esta etapa, vamos a explicar detalladamente cómo se diseñaron las diferentes interfaces usadas en el foro.

Interfaz Topic

Primero diseñaremos la interfaz designada como **Topic**, que mostrará todos los temas existentes en nuestra aplicación. Esta primera interfaz será una **Table View Controller**, en la que podremos diseñar una celda prototipo, que se usará para todas las filas.

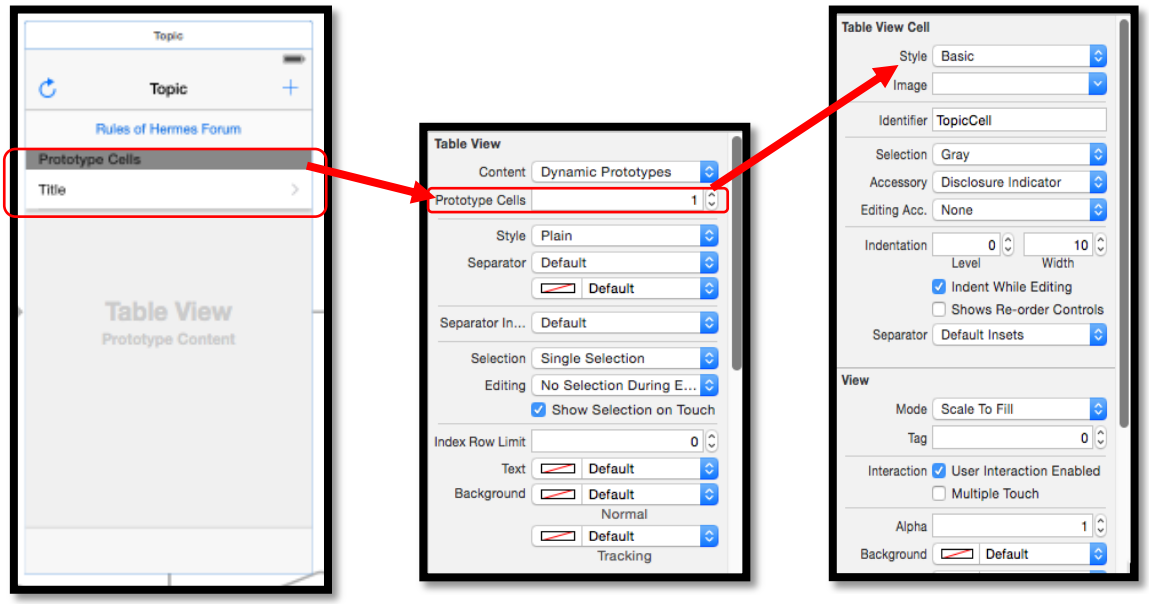


Ilustración 32. Diseño de la interfaz Topic.

Para crear la celda, seleccionamos la **Tabla View** y ponemos a uno **Prototype Cells**, después seleccionamos la celda y le indicamos que el valor de **Style** será **Basic**, de esta forma diseñaremos una celda sencilla con las características ofrecidas.

Después le añadimos un identificador (**IMPORTANTE:** este nombre nos servirá para identificar la celda en la parte de implementación) y cambiamos el valor de **Accesory** poniendo la opción de **Disclosure Indicator**, que mostrará un título y la flechita.

Por último, le añadimos 3 botones, uno dentro de la tabla y dos en la cabecera.

- Botón **Rules of Hermes Forum**, cuyo propósito será mostrar las reglas del foro. Este botón estará enlazado con la interfaz **Rules of Hermes Forum**.
- Botón **Add**, al pulsarlo nos enviará a la pantalla de añadir temas. Este botón estará enlazado con la interfaz **New Topic**.
- Botón **Refresh**, cuando lo pulsemos actualizará el contenido de la tabla.

Interfaz Rules of Hermes Forum

Nuestro segundo paso será diseñar la interfaz designada como **Rules of Hermes Forum**, que mostrará las diez reglas de la Nettiqueta.

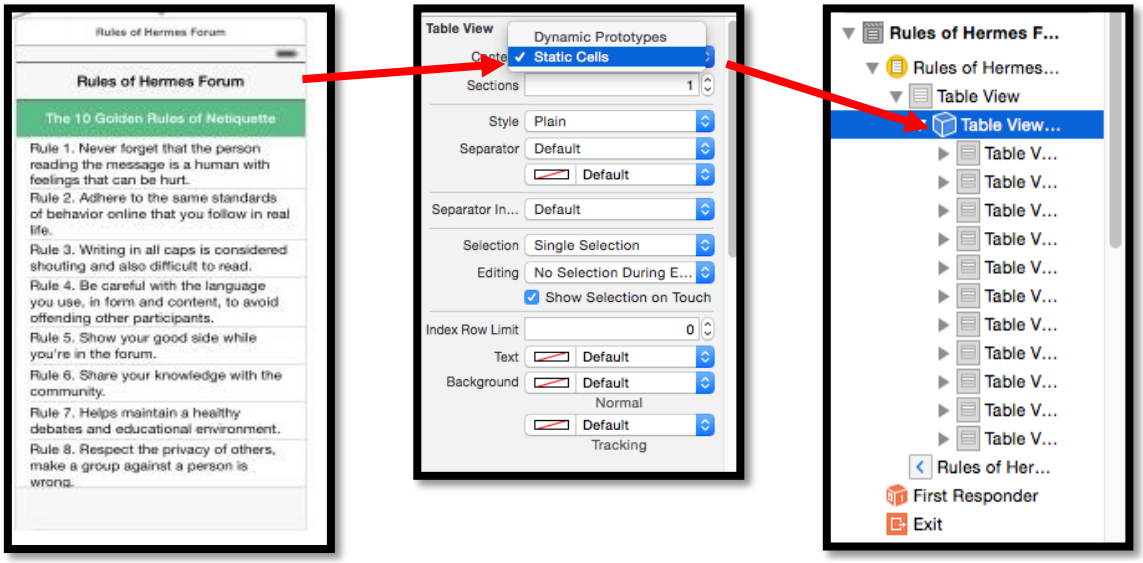


Ilustración 33. Diseño de la interfaz Reglas.

Esta será también una **Table View Controller**, donde cambiaremos el valor de **Content** de **Dynamic Prototype** a **static cell**, de esta forma le podemos añadir 11 filas, donde poner un título y las diez reglas.

Interfaz New Topic

En nuestro tercer paso diseñaremos la interfaz designada como **New Topic**, que nos permitirá crear un nuevo tema y añadirle el primer comentario.

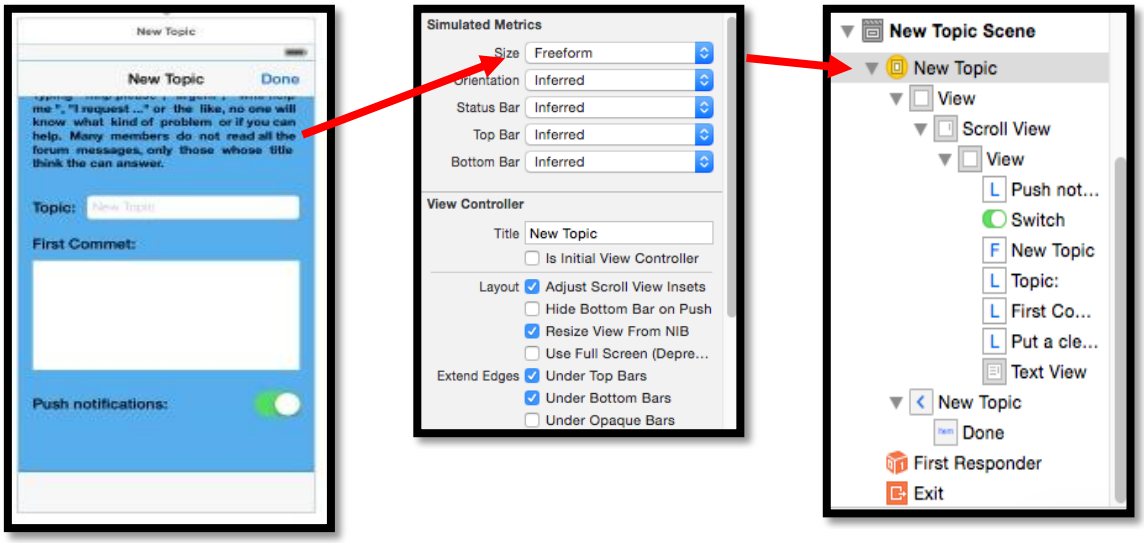


Ilustración 34. Diseño de la interfaz New Topic.

Esta interfaz será un **View Controller** de base, pero le añadiremos un **Scroll view**, para que todo su contenido pueda ser desplazado hacia arriba o abajo. Dentro del scroll añadiremos 4 **Labels**, uno de los cuales explicará detalladamente cómo crear un tema nuevo, un **Text Field** para poner el nombre del tema y por último un **Text View**, que no tiene límite de tamaño, siendo idóneo para escribir el comentario.

Para que el scroll funcione, primero tenemos que cambiar el valor de **Size** a **Freeform**, de esta forma no tendrá tamaño fijo.
Por último le añadiremos dos botones, uno en la cabecera y otro en la parte baja del interfaz.

- Botón **Done**, que enviará los datos al servidor de Parse, donde serán recogidos en sus tablas correspondientes.
- Botón **Notifications**, que activará o desactivará el uso de notificaciones.

Interfaz New Comment

El cuarto paso será casi idéntico al anterior, donde diseñaremos la interfaz designada como **New Comment**, que nos permitirá añadir un comentario al tema en cuestión.

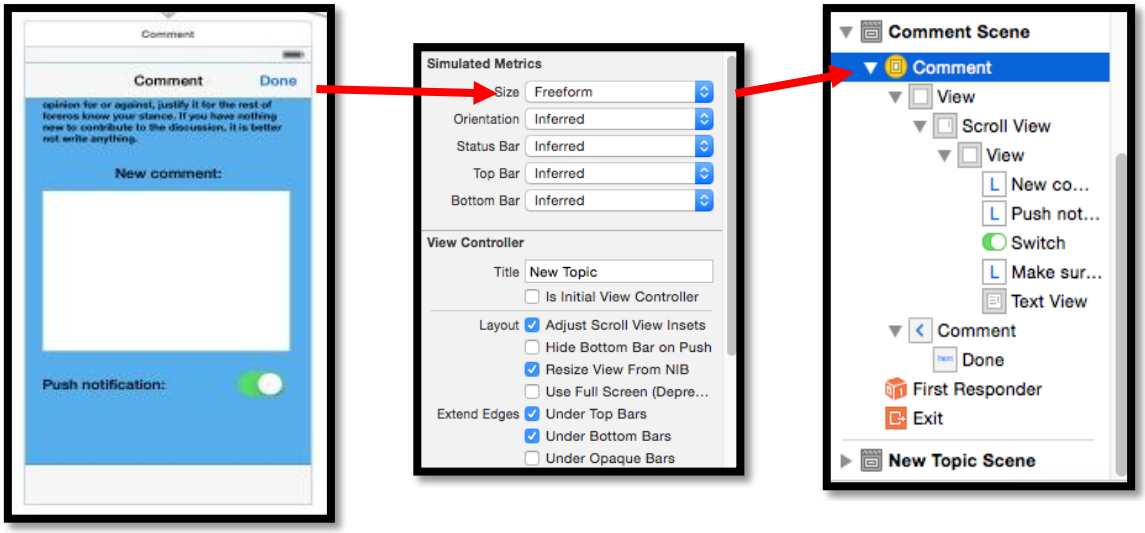


Ilustración 35. Diseño de la interfaz New Comment.

Esta interfaz será un **View Controller** de base, pero le añadiremos un **Scroll view**, para que todo su contenido pueda ser desplazado hacia arriba o abajo. Dentro del scroll añadiremos 2 **Labels**, uno de los cuales explicará detalladamente cómo añadir un comentario al tema y un **Text View** para añadir el comentario.
Por último le añadiremos dos botones, uno en la cabecera y otro en la parte baja del interfaz.

- Botón **Done**, que enviará los datos al servidor de Parse, donde serán recogidos en su tabla correspondiente.
- Botón **Notifications**, que activará o desactivará el uso de notificaciones.

Interfaz Comment

Nuestro quinto paso será diseñar la interfaz designada como **Comment**, que nos enseñará todos los comentarios relacionados con un tema en cuestión. La interfaz **Comment** estará enlazada con la celda prototipo del interfaz **Topic**.

Esta interfaz será una **Table View Controller**, en la que podremos diseñar una celda prototipo, que se usará para todas las filas.

Para crear la celda, seleccionamos la **Tabla View** y ponemos a uno **Prototype Cells**, después seleccionamos la celda y le indicamos que el valor de **Style** será **Custom**, de esta forma podremos diseñar el contenido de la celda.

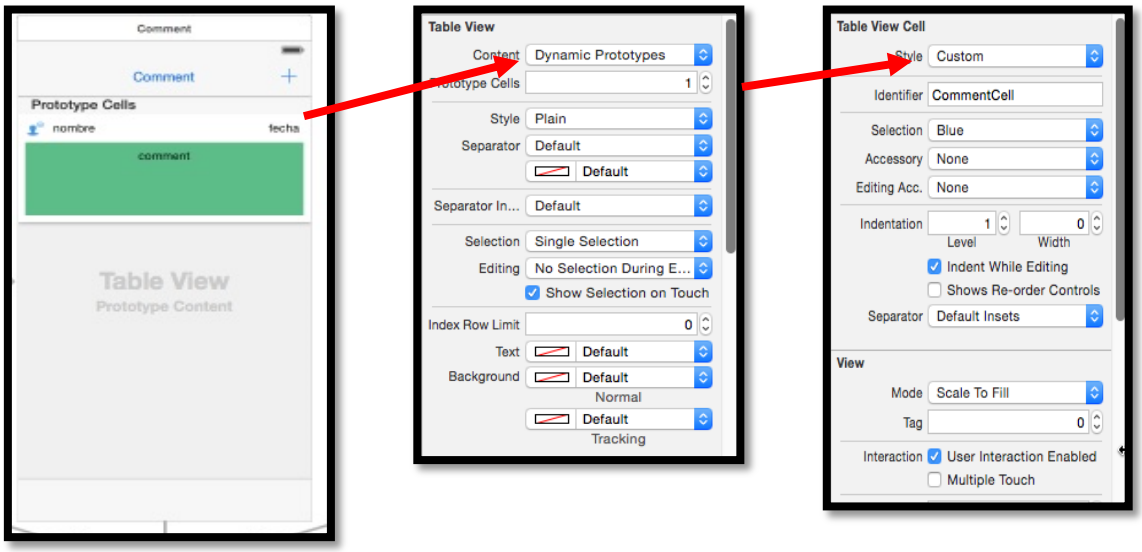


Ilustración 36. Diseño de la interfaz Comment.

Dentro de la celda añadiremos 2 Labels, uno indicará el nombre del usuario que dejo el comentario y otro la fecha, un identificador (**IMPORTANTE:** este nombre nos servirá para identificar la celda en la parte de implementación) y por último, le añadiremos un **Text View** para mostrar el comentario.

Esta interfaz tendrá 3 botones, uno dentro de la celda y dos en la cabecera.

- Botón **Invitar**, este botón tendrá una imagen y no título, al pulsarlo nos enviará a la pantalla de **AddFriend**.
- Botón **Add**, cuyo propósito es enviarnos a la pantalla de añadir comentarios. Este botón esta enlazado con la interfaz **New Comment**.
- Botón **Comments**, cuando lo pulsemos actualizará el contenido de la tabla.

Interfaz Add Friends

En el último paso de este punto, diseñaremos la interfaz designada como **Add Friends**, que nos ayudará a enviar un mensaje de correo a un desconocido.

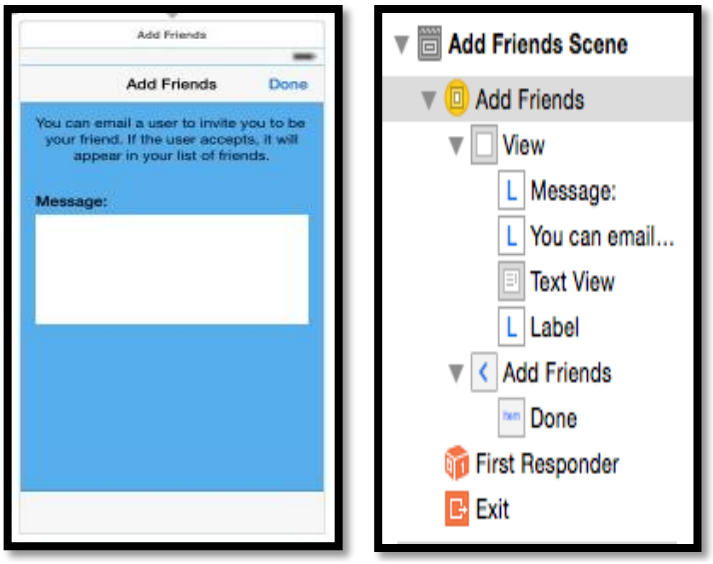


Ilustración 37. Diseño de la interfaz Add Friends.

Esta interfaz será un simple **View Controller**, que contendrá dos **Labels**, un **Text View**, para poder escribir un mensaje sin tener límite de palabras y un **Text Field**, que nos mostrará un mensaje indicando si el correo electrónico fue enviado.

4.3 BASE DE DATOS

En esta tercera etapa de la práctica, usaremos los servicios de Parse para crear nuestro Back-end, donde la aplicación podrá conectarse a un servidor remoto para guardar datos, recibir datos o archivos, incluso para hacer login e identificarte en la App.

Parse es lo que se denomina un Back-end as a service (**BaaS**), que nos va a permitir disfrutar de múltiples funcionalidades en la nube y ayudar a desarrollar aplicaciones que requieran un back-end con muy poco esfuerzo.

Ilustración 38. Base de datos.

4.3.1 FASE 1: REGISTRO EN PARSE

Ve a Parse y presiona **create an account**:

Ilustración 39. Registro de Parse.

Rellena todos los campos y ya estás dentro. También puedes pulsar **Sign up** para darte de alta en el servicio con Facebook o Github.

Crear una App

Ahora para interactuar con la plataforma y poder enviar notificaciones, necesitamos crear una aplicación dentro de Parse, eso lo hacemos dando clic en la pestaña que dice **App Keys**, si no tenemos ninguna aplicación creada, veremos un botón indicándote como iniciar el proceso.

Ilustración 40. Formulario de Parse.

La primera vez nos pedirá que terminemos de completar un formulario para el perfil y de paso crear la primera aplicación, después nos mostrará el **Application ID** y el **Client Key**, que serán necesarios para el desarrollo de la aplicación.

Ilustración 41. App Keys de Parse.

Las próximas veces que entremos en Parse, veremos la pantalla de inicio, donde se encuentran todas las app creadas.

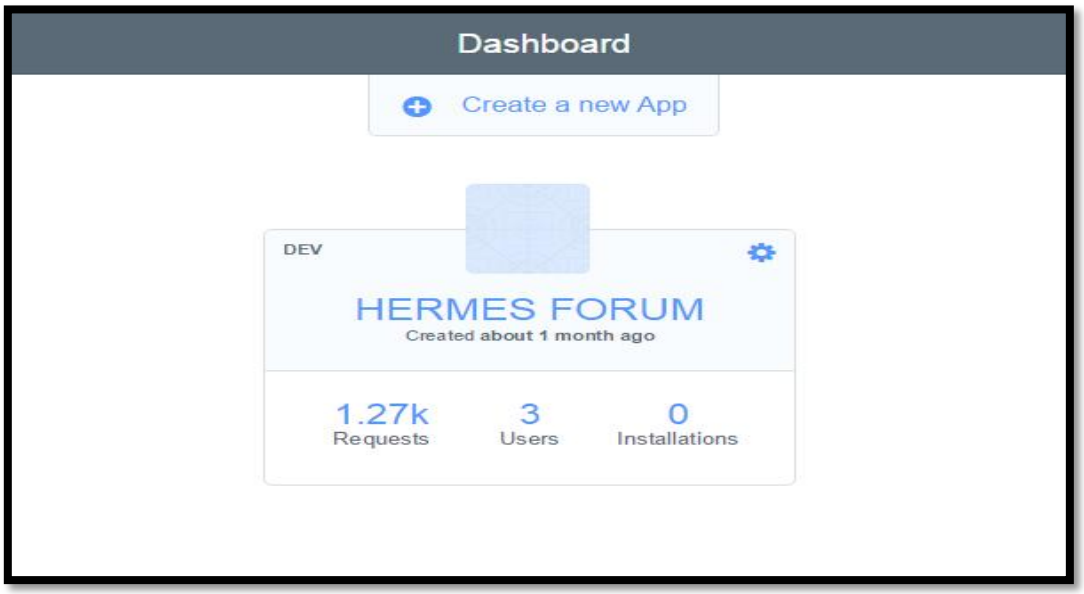


Ilustración 42. Página de inicio de Parse.

4.3.2 FASE 2: DISEÑO DE TABLAS

Si apretamos en una de las aplicaciones, nos mostrará la pantalla de Analytics, donde está marcada por defecto la pestaña de Audience, que muestra la cantidad de activaciones diarias de tu aplicación.

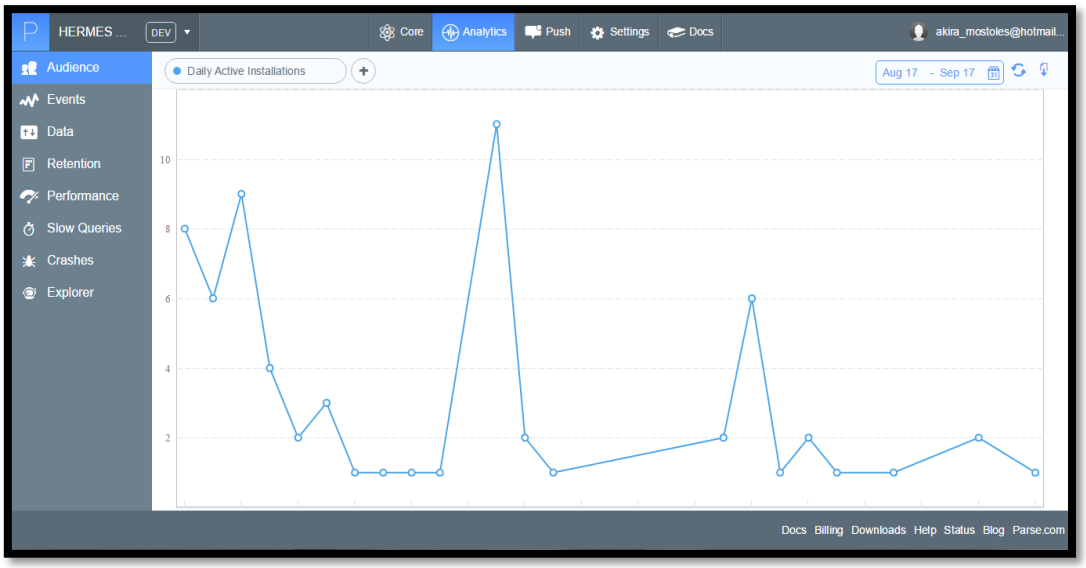


Ilustración 43. Opciones de Analytics.

A la izquierda tenemos el panel de herramientas, que son todos los servicios que ofrece Analytics, para analizar el funcionamiento de la aplicación.

Arriba en el centro tenemos el panel de Parse:

- **Core:** Todo lo que tu aplicación necesita para poder almacenar y consultar los datos sin un solo servidor. Además, puedes programar tareas repetitivas para que trabajen en segundo plano.
- **Analytic:** Admite el seguimiento de cualquier dato o acontecimiento que ocurre en tu aplicación en tiempo real. Te permite entender el uso de aplicaciones, encontrar y solucionar los accidentes, y medir el crecimiento y la retención como nunca antes.
- **Push:** En eta pestaña puedes enviar notificaciones push a través de dispositivos y plataformas con facilidad. Aprovechando la consola Push para programar campañas con antelación.
- **Settings:** Donde podremos encontrar todos los datos de nuestro perfil de Parse, así como las Keys de nuestra aplicación.
- **Docs:** Donde podremos descargarnos el SDK de Parse, documentación, guías, ejemplos, etc.

Al igual que C2Call, es necesario descargar el SDK de Parse para introducir en nuestro proyecto las librerías precisas, esto se consigue desplazando y copiando los archivos indicados.

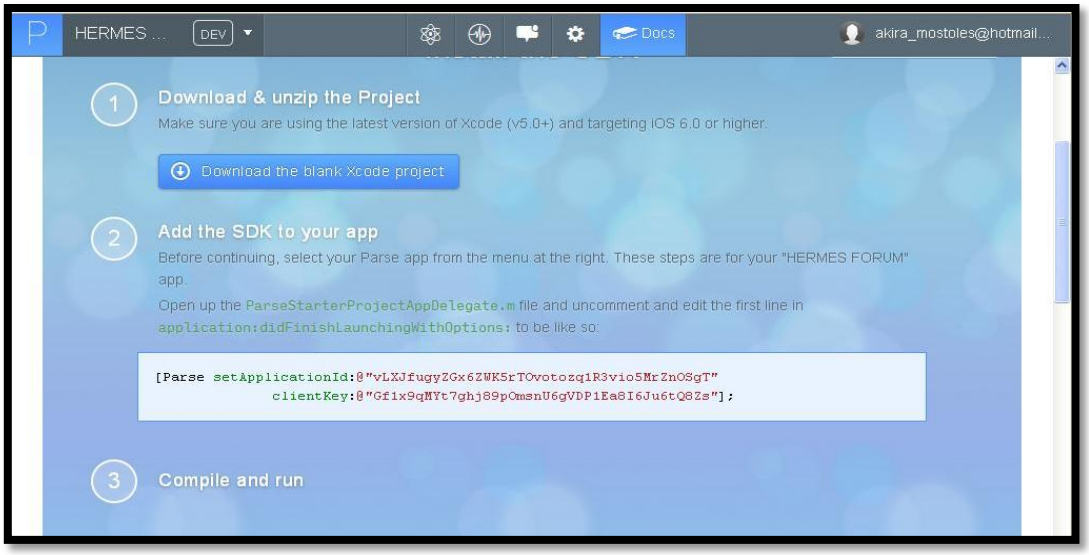


Ilustración 44. Pantalla de descarga de Parse.

También es necesario añadir dos líneas de código en el archivo **AppDelegate.m**, según nos indica la pantalla. Estas dos líneas son las app keys que establecerán la conexión con los servicios de Parse, pero para que la aplicación funcione correctamente, será necesario configurar una serie de permisos.

Crear mi primera tabla

Empecemos creando nuestra primera tabla, o **Class**, que así se llama en **Parse**.

Para ello nos iremos a la pestaña **Core** del panel de **Parse**, pulsaremos **Add Class**, e indicaremos un nombre para nuestra tabla. Antes de pulsar **Create Class** veréis que podemos elegir entre diferentes tipos de tabla. Esto, no es ni más ni menos, que para generar nuestra tabla en base a una plantilla.

Ilustración 45. Botón para crear tablas.

Una vez hecho esto, en la parte izquierda de la pantalla veremos que nos ha creado nuestra tabla. A continuación, vamos a crear una serie de campos para la misma. Para ello, pulsamos la opción **Add a row**.

Ilustración 46. Botón para crear filas.

Al terminar de crear las columnas, observamos que en pantalla no se ve ninguna de las columnas creadas. Eso es porque no hay ninguna fila en nuestra tabla. Para comprobarlo, pulsar la opción **+ Row** y veréis que la fila vacía que genera, no solo cuenta con los campos creados anteriormente, sino que también contiene alguno adicional que incluye **Parse** por defecto (objectId, createdAt, updatedAt y ACL).

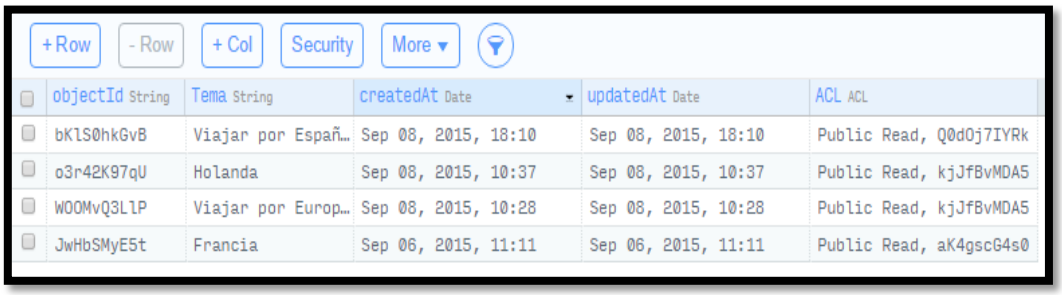
Para guardar todos los datos introducidos en el foro y los datos de registro, vamos a crear tres tablas.

1. Tabla de **Comentarios**, en esta tabla se guardará el **tema** al que pertenece el comentario, el **nombre** del usuario, el **comentario** y la **fecha** de cuando se escribió.

objectId String	Tema String	Nombre String	Comentario String	Fecha String	createdAt Date	updatedAt Date	ACL ACL
7HHaSh3R8t	Viajar por España	ALI	En España hay cientos de destinos turisti…	08/09/15	Sep 08, 2015, 18:10	Sep 08, 2015, 18:10	Public Read, Q0dOj7…
f33YWHV8dI	Holanda	LUCI	Recomiendo un crucero por los canales de …	08/09/15	Sep 08, 2015, 10:37	Sep 08, 2015, 10:37	Public Read, kjJfBv…
JMom1Awnme	Francia	LUCI	Recomiendo visitar Vezelay (Borgona). Es …	08/09/15	Sep 08, 2015, 10:32	Sep 08, 2015, 10:32	Public Read, kjJfBv…
RmeXVQ8H7L	Viajar por Europa	LUCI	La forma mas economica de ver Europa es u…	08/09/15	Sep 08, 2015, 10:28	Sep 08, 2015, 10:28	Public Read, kjJfBv…
cGhkkrjVUy	Francia	PRUEBA1	La mejor época para visitar Paris es otoñ…	05/09/15	Sep 06, 2015, 21:17	Sep 06, 2015, 21:17	Public Read, aK4gsc…
DWkJqZ1TZu	Francia	PRUEBA1	La Torre Eiffel es uno de los más destaca…	05/09/15	Sep 06, 2015, 11:11	Sep 06, 2015, 11:12	Public Read, aK4gsc…

Tabla 1. Tabla Comentarios.

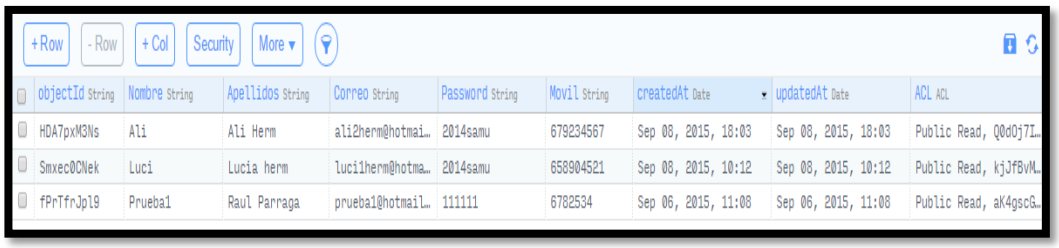
2. Tabla de **Temas**, en esta tabla se guardarán los temas creados en el foro.



objectId String	Tema String	createdAt Date	updatedAt Date	ACL ACL
bKlS0hkGvB	Viajar por Españ…	Sep 08, 2015, 18:10	Sep 08, 2015, 18:10	Public Read, Q0dOj7IYRk
o3r42K97qU	Holanda	Sep 08, 2015, 10:37	Sep 08, 2015, 10:37	Public Read, kjJfBvMDA5
WOOMvQ3LlP	Viajar por Europ…	Sep 08, 2015, 10:28	Sep 08, 2015, 10:28	Public Read, kjJfBvMDA5
JwHbSMyE5t	Francia	Sep 06, 2015, 11:11	Sep 06, 2015, 11:11	Public Read, aK4gscG4s0

Tabla 2. Tabla Temas.

3. Tabla **Persona**, donde guardaremos los datos de registro de cada usuario.



+Row | - Row | + Col | Security | More ▾

objectId String	Nombre String	Apellidos String	Correo String	Password String	Movil String	createdAt Date	updatedAt Date	ACL ACL
HDA7pxM3Ns	Ali	Ali Herm	ali2herm@hotmai…	2014samu	679234567	Sep 08, 2015, 18:03	Sep 08, 2015, 18:03	Public Read, Q0d0j7I…
SmxecOCNek	Luci	Lucia herm	luci1herm@hotma…	2014samu	658904521	Sep 08, 2015, 10:12	Sep 08, 2015, 10:12	Public Read, kjJfBvM…
fPrTfrJpl9	Prueba1	Raul Parraga	prueba1@hotmail…	111111	6782534	Sep 06, 2015, 11:08	Sep 06, 2015, 11:08	Public Read, aK4gscG…

Tabla 3. Tabla Persona.

Parse además crea automáticamente tres tablas por defecto.

- La tabla **User**, cada vez que un usuario se registra, guarda los datos principales, tales como username, password, email, etc. Es la tabla donde se comprueba si un usuario ya está registrado. Al no poder acceder a los datos de esta tabla, tuvimos que crear otra para guardar los datos de registro.
- La tabla **Session**, que nos indica cuanta gente hay conectada en la aplicación.
- La tabla **Role**, donde podemos crear usuarios especiales, que puedan modificar los datos de las tablas.

4.3.3 FASE 3: CLASES CREADAS

Para el desarrollo de la práctica hemos creados 8 clases diferentes, que contienen las variables, métodos y funcionalidades necesarias para el funcionamiento del foro.

Gracias a las APIs de C2Call, no fue necesario crear más clases para el funcionamiento del chat, llamadas VoIP, creación de grupos, agenda, etc. Pero como tenemos que añadir los servicios de Parse a la aplicación, será necesario modificar algunas de las clases de C2Call.

Para todas las clases de C2Call modificadas, se tuvo que incluir la librería de Parse:

```
#import <Parse/Parse.h>
```

SPLogoutViewController: Esta clase fue creada por C2Call y se encarga de desconectar al usuario. Fue necesario añadir una línea de código para desconectar al usuario de Parse.

```
[PFUser logOut]; //Código de Parse para desconectar al usuario
```

LoginViewController: Esta es otra de las clases creadas por C2Call, su función es la de comprobar si son correctos los datos de login consultado la información en el servidor. También fue modificada para comprobar los datos de Parse.

```
//Introducimos el usuario y el password en el login de Parse. El nombre de usuario lo pasamos a mayúsculas para evitar errores futuros, ya que Parse distingue entre mayúsculas y minúsculas.
 [PFUser logInWithUsernameInBackground: logUser.textContent.text.uppercaseString password: password.textContent.text block:^(PFUser *user, NSError *error)
{
    if (user) { //Si el login en Parse es correcto hacemos lo mismo en el de C2call.
      [[C2CallPhone currentPhone] loginWithUser: email.textContent.text  andPassword: password.textContent.text  withCompletionHandler:^(BOOL success, int resultCode, NSString *resultMessage){
         [pleaseWait hide];
          pleaseWait = nil;
         if (success) { // Si es correcto lanzamos la acción de login.
             if (loginDoneAction) {
                 loginDoneAction();
             }
          }else { // Si el login falla, mostramos un mensajes por el Prompt.
            [self showPrompt:NSLocalizedString(@"Login failed, invalid user name or
                                password!", @"Prompt")];
            [NSTimer scheduledTimerWithTimeInterval:3.0 target:self
                                selector:@selector(resetPrompt:)   userInfo:nil repeats:NO];
          }
       }];
      [[C2CallAppDelegate appDelegate] logEvent:@"UserLogin"];
    } else { // Si el login falla, mostramos un mensajes por el Prompt.
        pleaseWait = nil;
```

```
        [self showPrompt:NSLocalizedString(@"Login failed, invalid user name or password!",
                @"Prompt")];
        [NSTimer scheduledTimerWithTimeInterval:3.0 target:self
                selector:@selector(resetPrompt:) userInfo:nil repeats:NO];
    }
}];
```

SCRegistrationController2: Esta clase fue creada por C2Call, se encarga de enviar al servidor y de almacenar en la aplicación, todos los datos del registro. También fue modificada para enviar los datos a Parse.

```
//Primero crearemos un registro de usuario con el nombre, password y email. Que se guardara en la tabla User.
  PFUser *newUser =[PFUser user];
  newUser.email=self.email.textContent.text;
  newUser.username =self.firstName.text.uppercaseString; //El nombre de usuario lo pasamos a mayúsculas para evitar errores futuros, ya que Parse distingue entre mayúsculas y minúsculas. Este proceso se repite en el login.
  newUser.password =self.password1.textContent.text;

  //Segundo creamos un registro para guardar todos los datos que nos interesan en una tabla creada por nosotros.
  PFObject *miRegistro =[PFObject objectWithClassName:@"Persona"]; //La clase que representa un registro en una tabla de Parse es PFObject. Por lo tanto creamos el objeto para almacenar los datos y le indicamos que será en la tabla "Persona".
  [miRegistro setObject:self.firstName.text forKey:@"Nombre"]; //Le indicamos las columnas del registro que deseamos rellenar.
  [miRegistro setObject:self.lastName.text forKey:@"Apellidos"];
  [miRegistro setObject:self.email.textContent.text forKey:@"Correo"];
  [miRegistro setObject:self.password1.textContent.text forKey:@"Password"];
  [miRegistro setObject:self.phoneNumber.textContent.text forKey:@"Movil"];
```

NewTopicController: Esta clase fue creada para controlar y enviar a Parse, los datos utilizados en la creación de un nuevo tema. Comprueba que los campos no están vacíos y muestra una notificación de error en caso contrario.

```
if ([self.NewTopic.text isEqualToString:@""]) //Si el campo NewTopic está vacío sacamos un mensaje de aviso.
  {
    [self showPrompt:NSLocalizedString(@"Topic is mandatory field!", @"Prompt")];
    [NSTimer scheduledTimerWithTimeInterval:5 target:self selector:@selector(resetPrompt:)
userInfo:nil repeats:NO];
    return;
  }
```

ForumTableViewController: Esta clase fue creada para descargar y mostrar todos los temas de la aplicación. Tiene un botón que nos envía a la interfaz **Comments,** otro que nos muestra las reglas del foro y un tercero que sirve para actualizar la tabla.

```
//Creamos un MutableArray, que puede ser modificado durante la ejecución, para operar con la información recuperada de las tablas de Parse.
  NSMutableArray *content;
//Transforma la clase objeto en un String y se lo asigna a cadena.
  NSMutableString *cadena=[NSMutableString stringWithFormat:@"%@", object[i]];
```

```
    NSRange detectar = [cadena rangeOfString:@"Tema ="]; //Buscamos la posición inicial donde aparece "Tema" dentro del String.
    [cadena replaceCharactersInRange:NSMakeRange(0, detectar.location+7) withString:@""]; //Le decimos que remplace por "" desde la posición 0 hasta la posición donde se encuentra "Tema" + su longitud. De esta forma queda solo el contenido de Tema.
    NSCharacterSet *want = [NSCharacterSet characterSetWithCharactersInString:@"\";\n}"]; //Creamos un selector de caracteres.
    NSString *cadena2= [[cadena componentsSeparatedByCharactersInSet: want ]
    componentsJoinedByString:@""]; //Usamos el selector para quitar algunos caracteres de separación.
//Cambiamos el código de algunas letras para mejorar la compresión de los textos.
    cadena2 = [cadena2 stringByReplacingOccurrencesOfString:@"\\U00f1" withString:@"ñ"];
    NSString *vocalA = [cadena2 stringByReplacingOccurrencesOfString:@"\\U00e1" withString:@"á"];
    NSString *vocalE = [vocalA stringByReplacingOccurrencesOfString:@"\\U00e9" withString:@"é"];
    NSString *vocalI = [vocalE stringByReplacingOccurrencesOfString:@"\\U00ed" withString:@"í"];
    NSString *vocalO = [vocalI stringByReplacingOccurrencesOfString:@"\\U00f3" withString:@"ó"];
    NSString *vocalU = [vocalO stringByReplacingOccurrencesOfString:@"\\U00fa" withString:@"ú"];
```

CommentTableViewController: Esta clase fue creada para descargar y mostrar todos los comentarios relacionados con un tema. Este tema será enviado a través de un **segue** por la interfaz **Topic** y a su vez, reenviado a la clase **NewComment**.

```
- (void) prepareForSegue: (UIStoryboardSegue *) segue sender:(id)sender { // Esta función se encarga de comprobar el tipo de información que debe ser enviada por cada segue.

    if ([segue.identifier isEqualToString:@"AddFriend"]) { //Usamos el segue con identificador AddFriend

        MessageViewController *primero= [segue destinationViewController];
        NSString *nameR= [names objectAtIndex:[[self.tableView indexPathForSelectedRow]row]];
        primero.Nombre=nameR;
    }

    if ([segue.identifier isEqualToString:@"NewComment"]) { //Usamos el segue con identificador NewComment

        NewCommentViewController *segundo = [segue destinationViewController];
        segundo.Tema=Tema;
    }
}
```

NewCommentViewController: Esta clase fue creada para controlar y enviar a Parse, los datos utilizados en la creación de un nuevo comentario. Comprueba que los campos no están vacíos y muestra una notificación de error en caso contrario.
Para enviar un comentario a Parse, es preciso saber a qué tema pertenece. De esto se encarga la clase **Comment**, que enviará a través de un **segue** la información necesaria. Para recibir la información, es necesario que el **.h** tenga una variable creada.

```
@property (nonatomic, strong) NSString *Tema; //Creamos esta variable para recibir los datos enviados por el segue.
```

Después se necesitará añadir en el **.m** la referencia al archivo e inicializar la variable.

```
#import "NewCommentViewController.h"
@synthesize Tema,NComment; //Inicializa una variable.
```

MessageViewController: Esta clase fue creada para enviar un email a un conocido, recibiendo el nombre del destinatario a través de un **segue** de la interfaz **Comment**. Para poder enviar correos electrónicos utilizando el componente que ofrece iOS para ello, deberemos incluir el framework **MessageUI.framework** en nuestra aplicación, y además, hacer que nuestro view controller implemente el delegate correspondiente:

```
[MFMailComposeViewController canSendMail] //Comprobamos si nuestro dispositivo puede enviar mails en este momento.

MFMailComposeViewController *mailer = [[MFMailComposeViewController alloc] init]; //Iniciamos el controlador.

mailer.mailComposeDelegate = self;

[mailer setSubject:@"You want to be my friend?"]; //Asunto del mensaje.
NSString *emailBody = message.text; //Texto que incluirá el email que vamos a enviar.
[mailer setMessageBody:emailBody isHTML:NO]; //Muestra la vista del mail.
```

WUChatHistoryController: Esta clase fue creada por C2Call, se encarga de controlar las opciones de grupo (crearlos, borrarlos, elegir usuarios, foto de perfil, etc.) y enviarlos a favoritos (Agenda).

AppDelegate: Esta clase fue creada por defecto, pero es la más importante de todas. Su tarea principal es lanzar todos los métodos iniciales de la aplicación. Es decir, comprueba que existe conexión a internet, se registra con los servicios de Parse y C2Call, utilizando los **Application ID** y el **Client Key** obtenidos en el registro, y en caso de no poder registrarse muestra un mensaje de error.

```
[Parse enableLocalDatastore]; //Permite el almacenamiento y consulta de datos. Comenta esta línea si no desea usar las funciones de almacenamiento y prefiere usar cachePolicy.

[Parse setApplicationId: APPLICATION_ID clientKey: CLIENTE_KEY]; //Registra la aplicación con nuestra app registrada en Parse.

[PFAnalytics trackAppOpenedWithLaunchOptions:launchOptions]; //Envía una notificación a Parse para indicar que la hemos abierto en el simulador.

[PFUser enableAutomaticUser]; //Activa el usuario automático, para que se pueda trabajar con un usuario desde el momento que se inicia la aplicación, sin necesidad de una solicitud de red.

PFACL *defaultACL = [PFACL ACL]; //La clase PFACL se utiliza para controlar que los usuarios puedan acceder o modificar un objeto en particular.

// Si quieres que todos los objetos sean privados por defecto, comenta esta línea.
[defaultACL setPublicReadAccess:YES];

[PFACL setDefaultACL:defaultACL withAccessForCurrentUser:YES];
//Si ponemos YES, el PFACL que se aplica a la instancia recién creada de PFObject, proporcionará los accesos de lectura y escritura en el momento de su creación.
```

- **Si ponemos NO, se utilizará sin modificación el acl.**
- **Si acl es nil, este valor se ignora.**

4.3.4 DISEÑOS FINALES

Los diseños finales de cada apartado de la aplicación son:
Diseño de **Login/Register**:

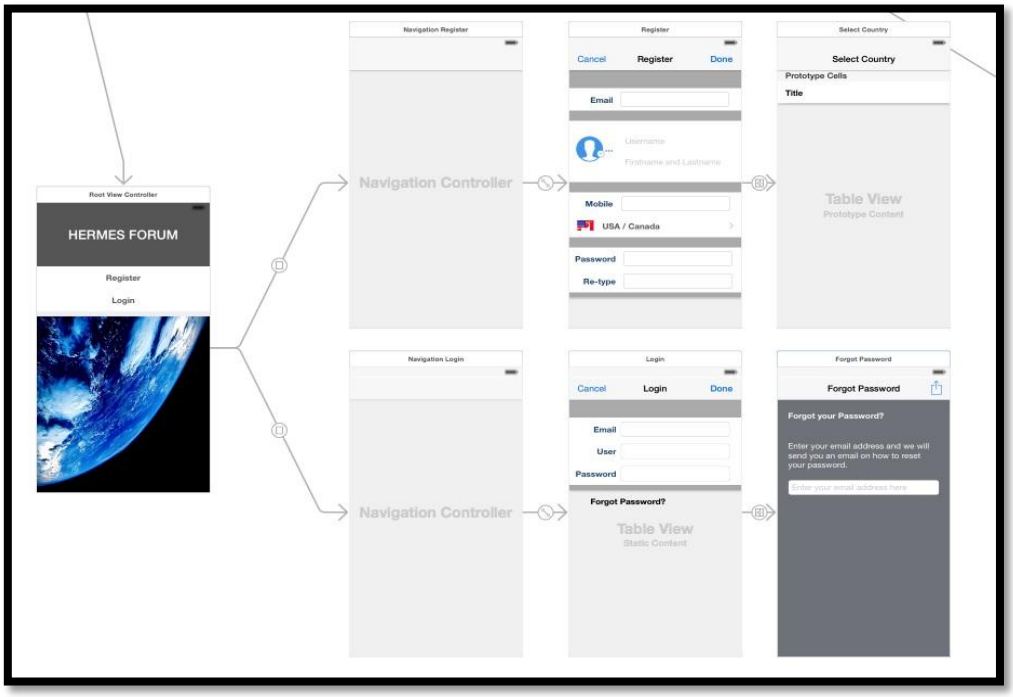


Ilustración 47. Diseño final Login/Register.

Diseño de **More** (Mas):

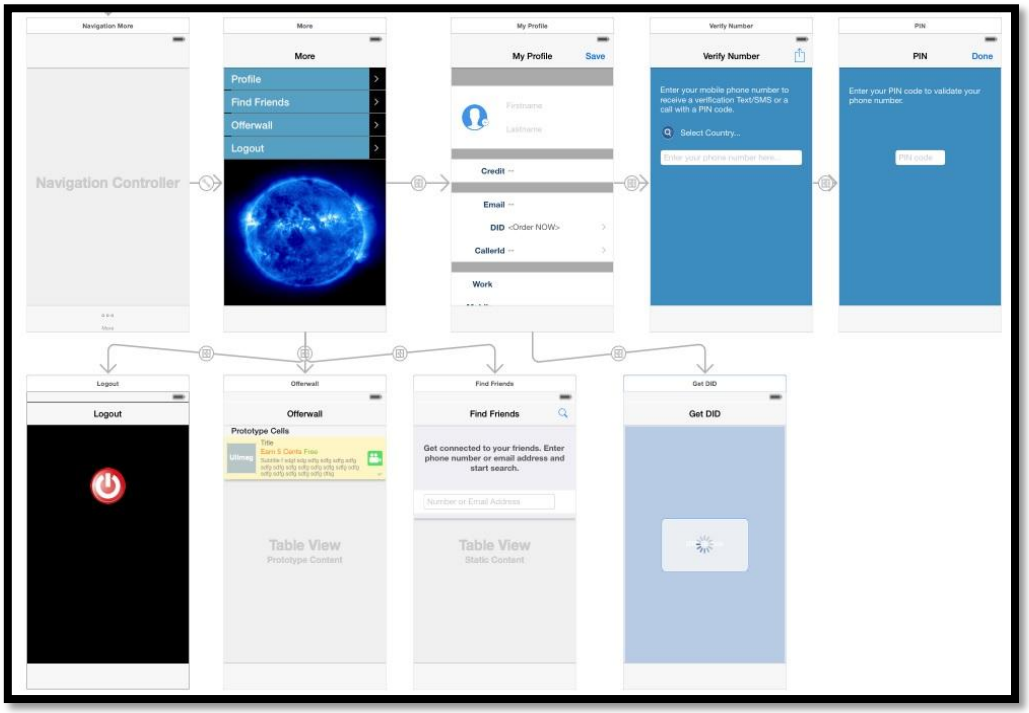


Ilustración 48. Diseño final More.

Diseño de **Friends** (Agenda):

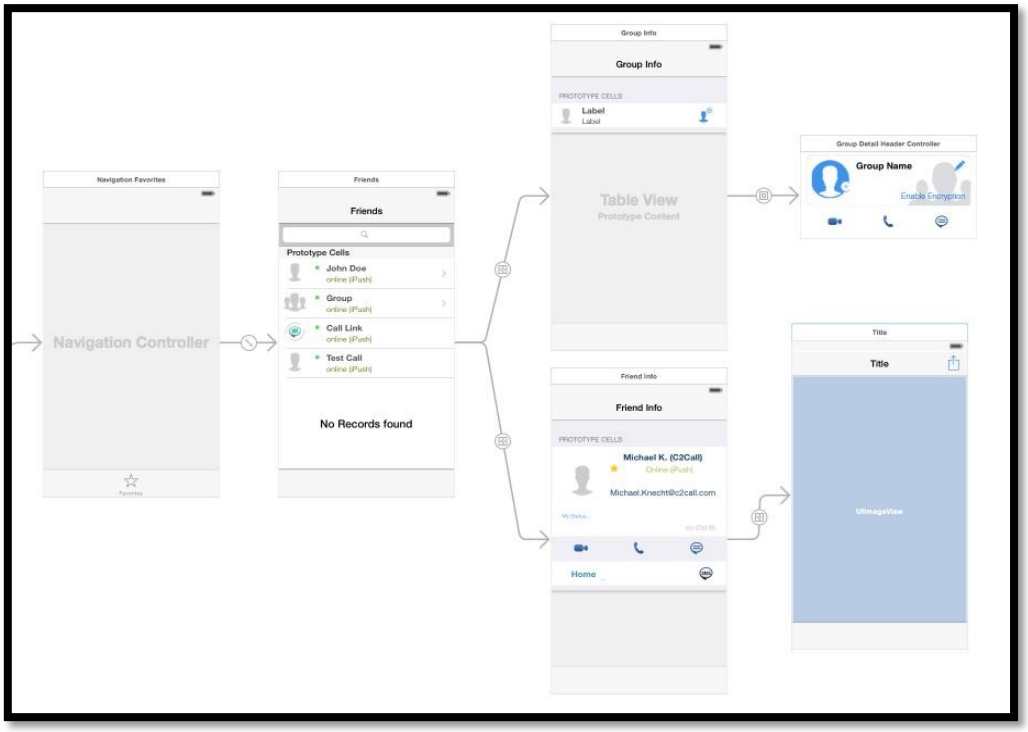


Ilustración 49. Diseño final Friends.

Diseño de **Recents** (Recientes):

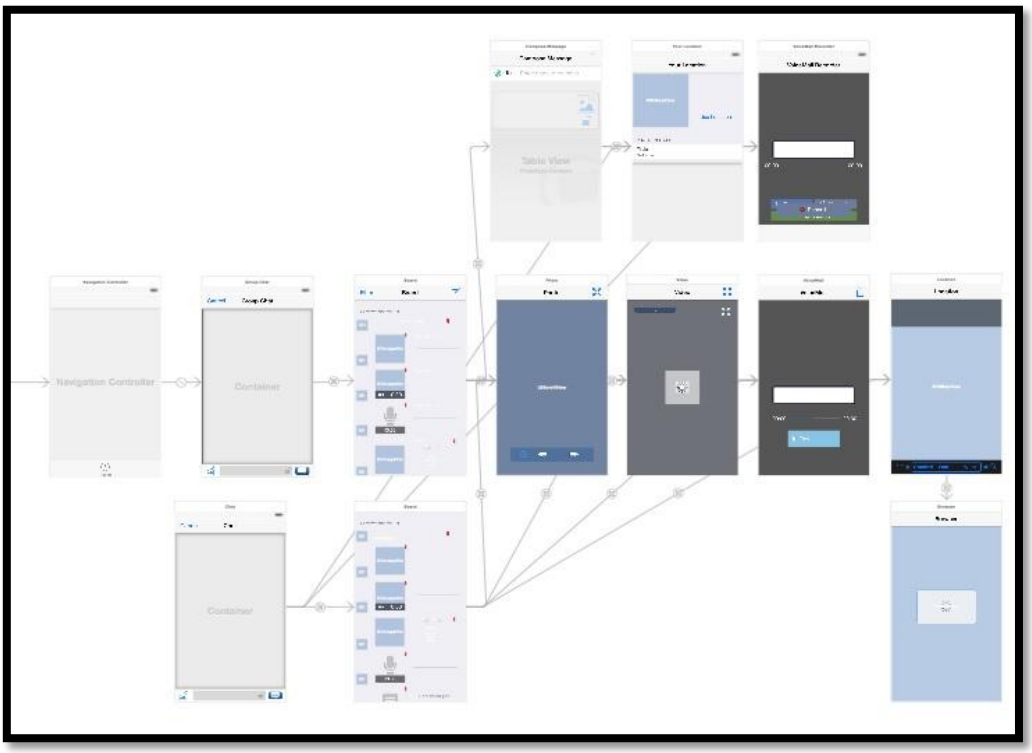


Ilustración 50. Diseño final Recents.

Diseño de **Chats**:

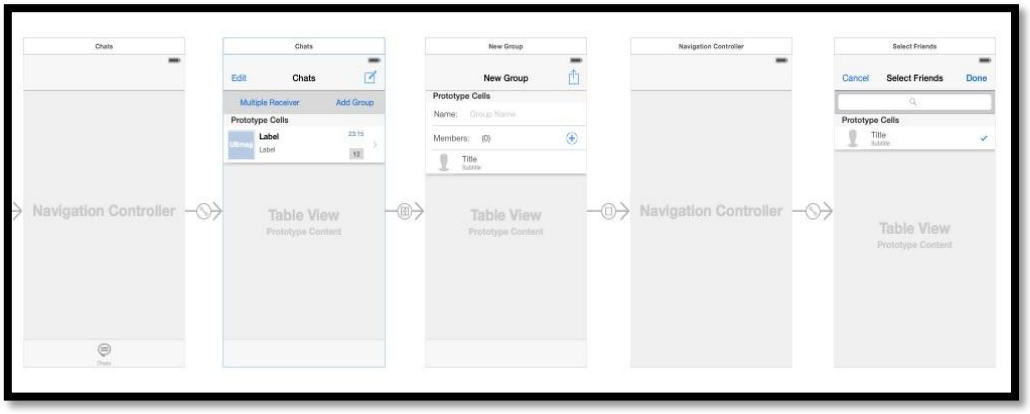


Ilustración 51. Diseño final Chats.

Diseño del **Foro**:

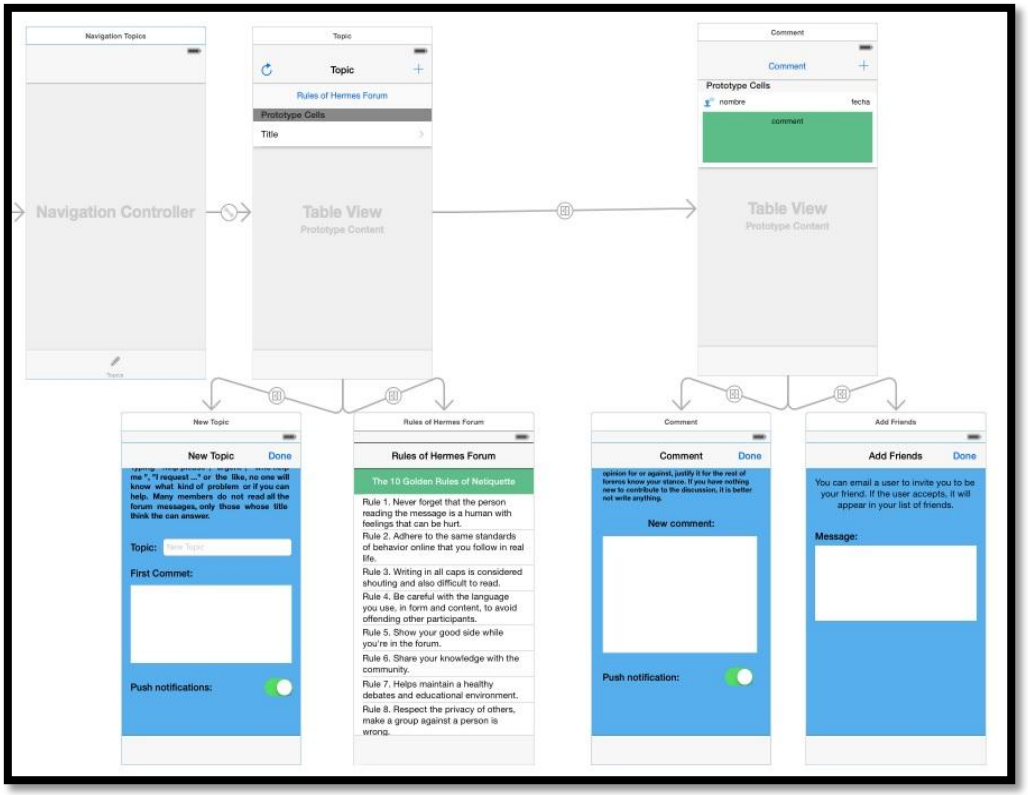


Ilustración 52. Diseño final Foro.

Si se observa detalladamente, las interfaces de **New Topic** y **Comennt** parecen tener cortado el texto de cabecera. Esto es debido a que se utilizó un **Scroll view** que puede desplazar hacia arriba o abajo el contenido, pero al ejecutar la aplicación se podrá comprobar que el contenido se muestra correctamente.

Diseño Completo de la **Aplicación**:

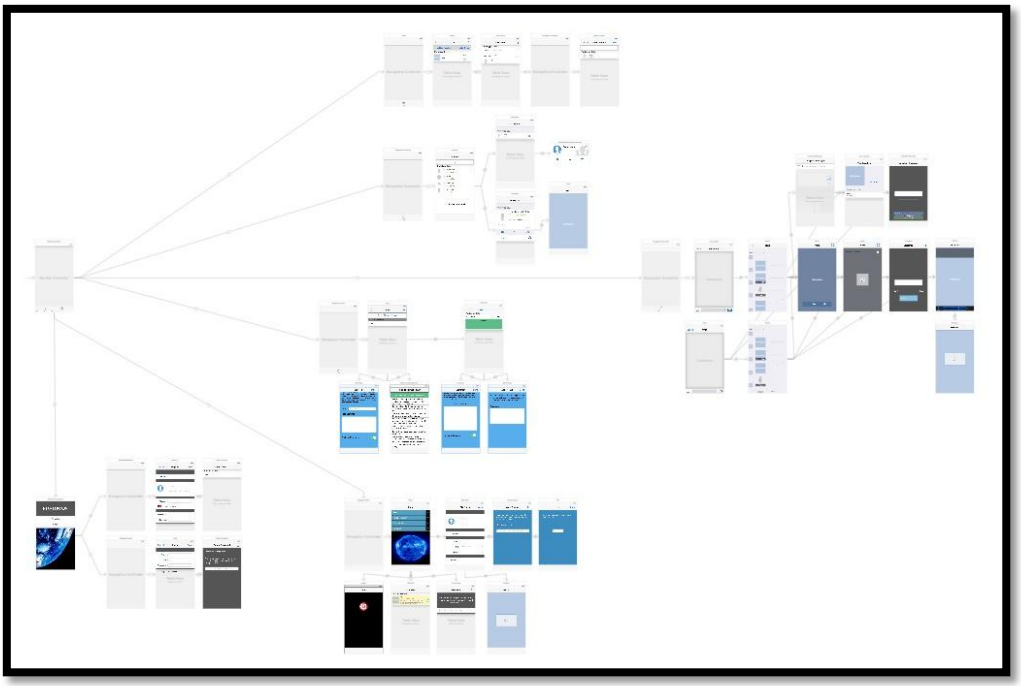
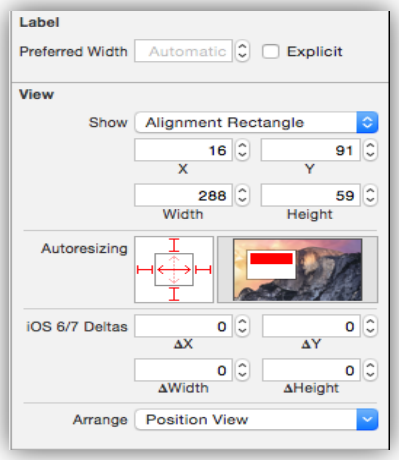


Ilustración 53. Diseño final de la aplicación.

4.4 COMPATIBILIDAD DE DISEÑOS

En esta última etapa del proyecto, vamos a verificar que todas las interfaces diseñadas se vean correctamente en cada uno de los diferentes iPhone. Para ello usaremos el Simulador iOS, que nos permitirá ejecutar la aplicación en un iPhone 4s, iPhone 5, iPhone 5s, iPhone 6 y iPhone 6s.

Nada más ejecutar la aplicación en el iPhone 4s, fue visible que el diseño no era compatible con los diferentes tamaños. Para poder hacer compatible las interface con cada uno de los iPhone, fue necesario usar las propiedades de **Autoresizing** ofrecidas por xcode.



Para realizar este proceso de **Autoresizing**, tendremos que modificar uno a uno todos los elementos de cada interfaz. Por suerte, la mayoría de nuestras APIs son **Table View Controller**, que tienen ajustado el tamaño por defecto

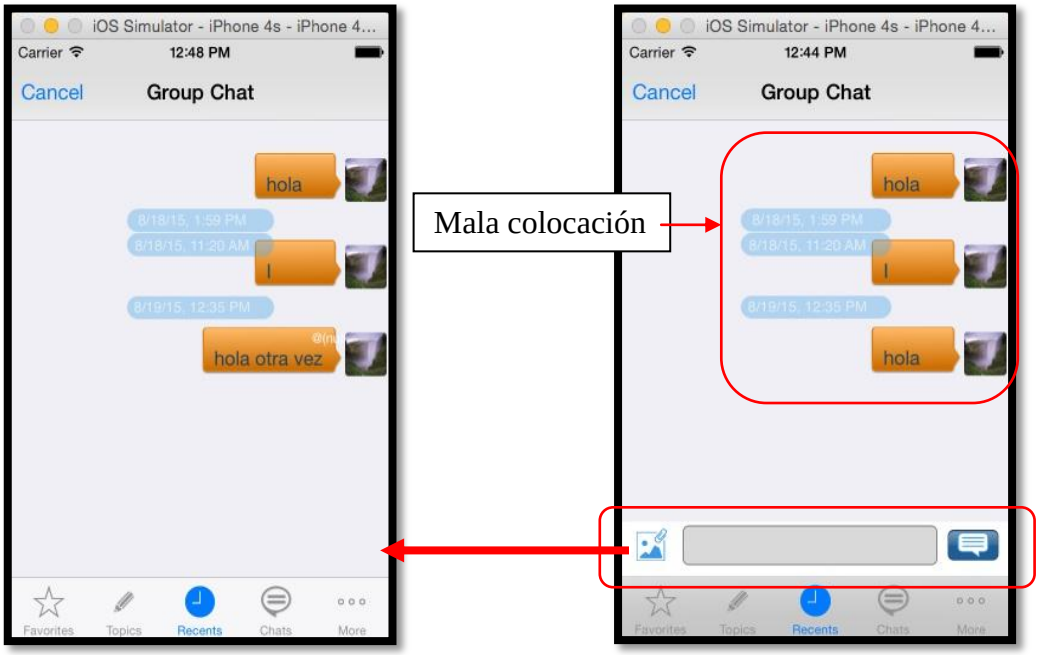


Ilustración 54. Ejemplo de diseño incompatible para todo los iPhone.

Una vez terminado el proceso de ajuste, se vería así:

Ilustración 55. Ejemplo de diseño compatible para todos los iPhone.

Esta forma de ajustar el tamaño es posible porque tenemos la opción de **Auto Layout** desconectada en el Main.storyboard. Si la opción estuviera activada, se tendría que crear un contenedor para cada interfaz, que contendría el tamaño de cada elemento, distancia entre cada uno, márgenes, etc.

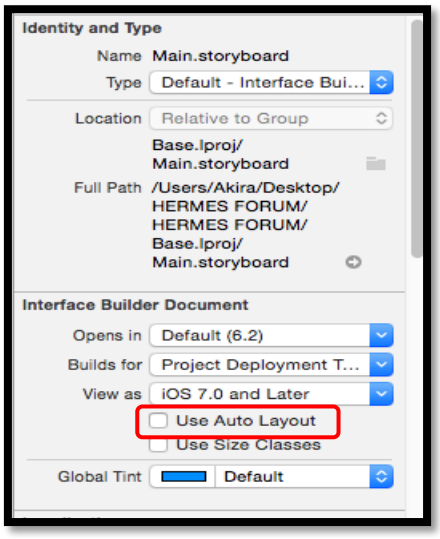


Ilustración 56. Opción de Auto Layout desconectada.

Una vez solucionado el problema de diseño, comprobaremos la visualización del segundo idioma. Para introducir un segundo idioma en la aplicación, basta con activar

la casilla del idioma deseado. Seleccionamos el archivo **Main.storyboar** y en la parte derecha de xcode nos saldrá la opción de añadir hasta 6 idiomas más.

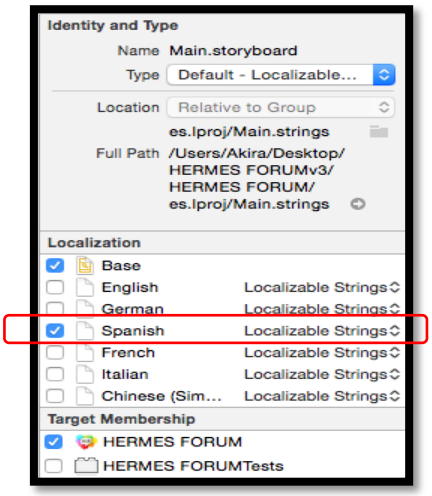


Ilustración 57. Opciones de idioma.

Al activar la casilla, el xcode nos creará un archivo llamado **Main.strings(Spanish)**, donde se mostrarán todos los Label, TextView, TextField, etc. de la aplicación.
Al abrir el archivo podremos distinguir dos líneas, una línea verde, con el tipo de archivo, su ID y el texto en inglés, y una línea roja, donde tendremos que añadir su traducción.

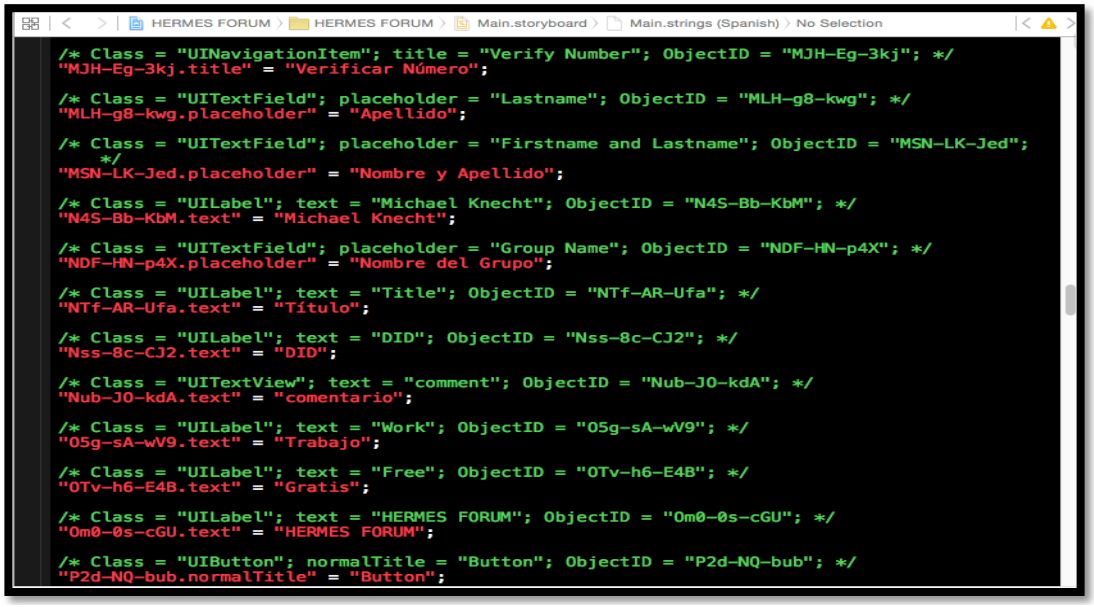


Ilustración 58. Archivo Main.strings(Spanish).

Una vez traducido el documento, ejecutamos la aplicación en español para comprobar que se visualiza correctamente. Algunos textos serán más grandes en español, por eso es necesario ajustar los márgenes y agrandar el campo.

El último paso, para terminar de introducir el segundo idioma, será la traducción de los mensajes mostrados por el Prompt y los avisos o notificaciones integradas en la app.

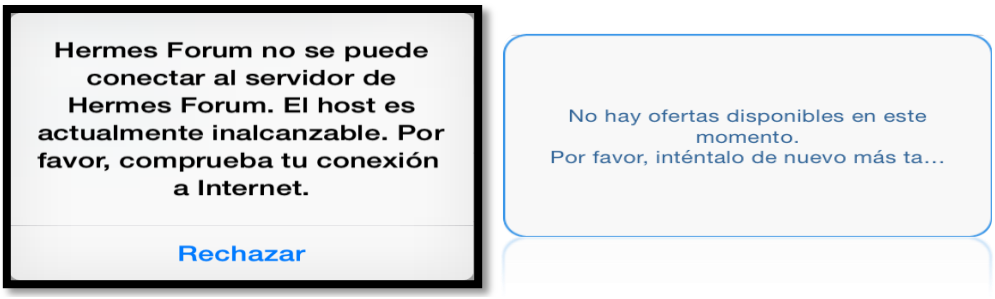


Ilustración 59. Notificaciones internas de la App.

En nuestro caso, el prompt es una consola que mostrará al usuario un mensaje indicado con anterioridad, esta se encuentra en la parte superior de nuestro iPhone.

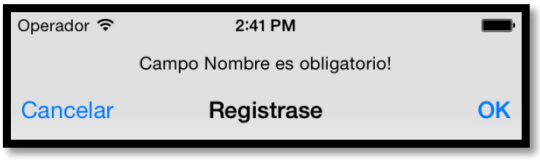


Ilustración 60. Prompt del simulador iOS.

Los pasos a seguir para traducir estos archivos son:

1. Haciendo clic en el proyecto, buscamos la sección "Localizations" que aparece a la derecha de xcode. Tienes que añadir los idiomas que vaya a soportar la aplicación (por ejemplo inglés y español).

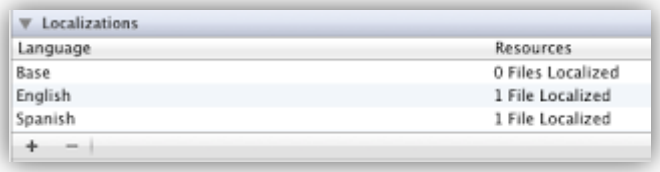


Ilustración 61. Localizations.

2. Tienes que añadir al proyecto un nuevo archivo de recursos de tipo **Strings** y lo llamas **Localizable.strings**.

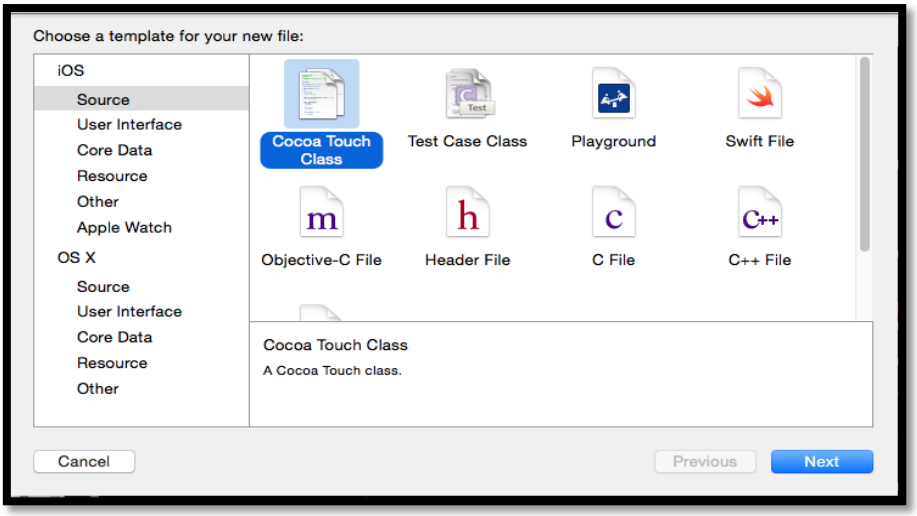


Ilustración 62. Pantalla para crear un archivo.

3. Seleccionamos el nuevo archivo y en el lado derecho, bajo la descripción del archivo, aparece la opción Localizar. Lo localizamos y añadimos los idiomas de inglés y español.

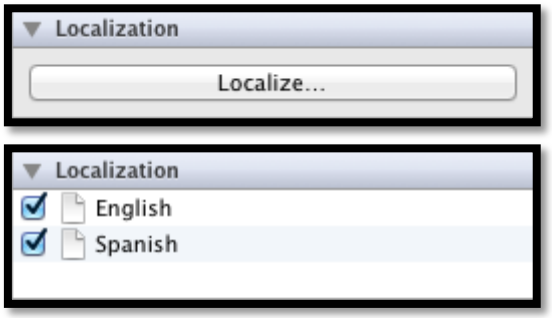


Ilustración 63. Añadir idiomas a un archivo.

4. Lo que Xcode realmente hace es crear dos carpetas (**en.lproj** y **es.lproj** para inglés y español respectivamente) donde crea dos versiones diferentes del archivo **Localizable.strings**. Luego en tiempo de ejecución detecta el idioma en el que el usuario tiene configurado su dispositivo y carga la correcta.

5. Abrimos uno de los dos archivos y el formato que hay que seguir para añadir traducciones es: **"Key" = "Value";** es decir una clave entre comillas dobles, símbolo igual, y el valor (normalmente la traducción) también entre comillas, y por último el punto y coma al final.

```
"KEY" = "Value";

  // Valor por defecto --> Valor localizado
"Hello" = "Hola";

  // Key que identifica al elemento --> Valor localizado
"Button_Play" = "Jugar";
```

5. PRUEBAS

Una vez que se ha generado el código, comienza las pruebas del software o sistema que se ha desarrollado. De acuerdo con Pressman, el proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la ejecución de pruebas para la parte aplicativa del mismo.

Para asegurar el correcto funcionamiento de la aplicación y garantizar la satisfacción del cliente, se debe implementar una estrategia de pruebas que incluya:

- Pruebas unitarias: individuales de cada método.
- Pruebas de integración: en conjunto.
- Pruebas del sistema: compatibilidad con requisitos.
    - Funcionalidad: cumple requisitos.
    - Rendimiento
    - Instalación: en la plataforma HW/SW de operación

Durante la etapa de diseño e implementación, se han realizado diferentes pruebas unitarias, para comprobar el funcionamiento de los métodos implementados, y pruebas de integración, para comprobar la funcionalidad de los diseños. En esta última etapa del proyecto, nos falta solo realizar las pruebas del sistema, para comprobar que la aplicación cumple con los requisitos.

Pruebas de Funcionalidad

- En la primera prueba dejaremos campos libres en la pantalla de Registrarse, para probar si la aplicación nos avisa del error, y registraremos a un usuario, para asegúranos de que los datos son guardados correctamente.

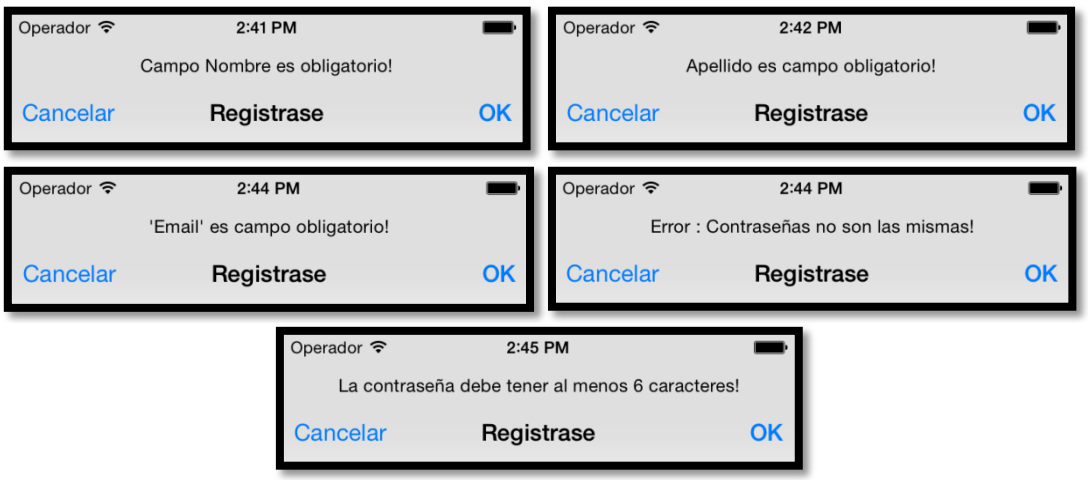


Ilustración 64. Pruebas en la pantalla de Registro.

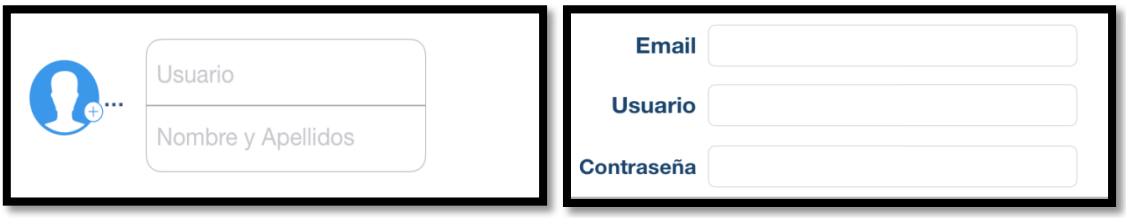
Se detectó correctamente los campos vacíos y al terminar el registro, los datos fueron guardados correctamente en los servidores de Parse y C2Call.

- Como segunda prueba, realizaremos otro registro, pero usando el mismo usuario. Para comprobar que C2Call nos avisa de que el usuario ya está registrado.

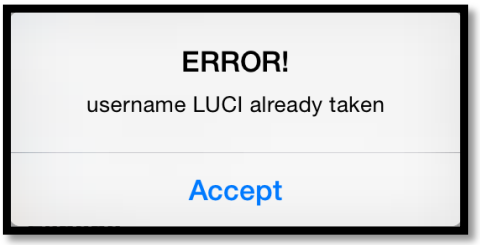
En esta segunda prueba, descubrimos que el registro no indica ningún error y entra en la aplicación. Después de investigar la causa, averiguamos que el problema reside en los servicios de C2Call y Parse, es decir, cada uno utiliza un campo diferente como usuario. C2Call utiliza el correo electrónico, pero Parse solo el nombre y además distingue entre mayúsculas y minúsculas.

Para solucionar este problema, tuvimos que cambiar los campos de registro, sustituyendo el campo de Nombre por Usuario, y el campo Apellido, por Nombre y Apellidos.

Además tuvimos que añadir un campo más en la pantalla de login, pidiendo ahora el Email, Usuario y Contraseña.



Al repetir la prueba, la aplicación detecta correctamente el usuario repetido y nos muestra un mensaje.



- En esta tercera prueba, realizaremos los mismos pasos para la pantalla de login, comprobando además si detecta la introducción de una contraseña errónea.

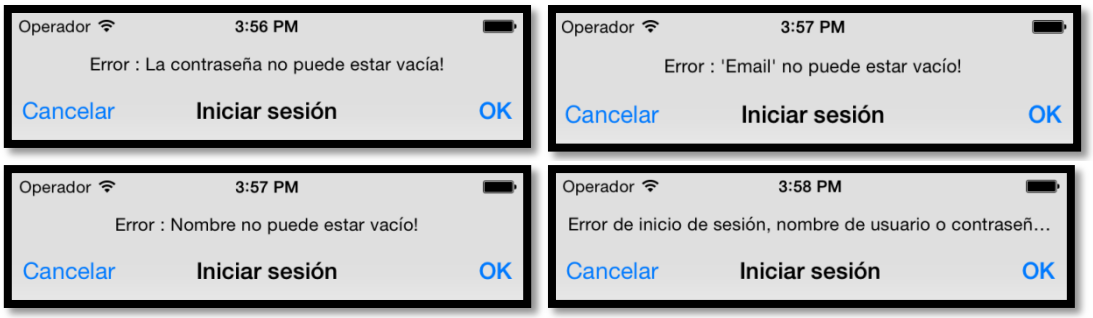


Ilustración 65. Pruebas en la pantalla de Inicio.

Se detectaron correctamente los campos vacíos y nos avisó de contraseña errónea.

- En la cuarta prueba, vamos a comprobar si la interfaz de Tema Nuevo, es capaz de detectar que sus campos están vacíos, que el nombre del tema llega a los 6 caracteres y si el tema introducido ya existe.

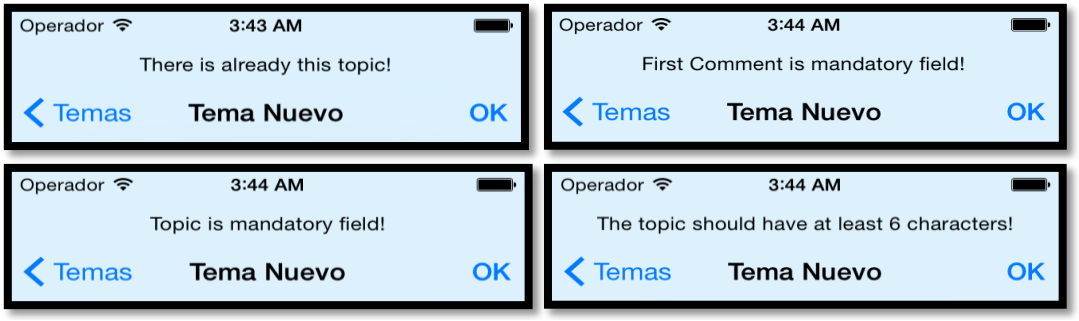


Ilustración 66. Pruebas en la pantalla de Tema Nuevo.

Se detectó correctamente todos los errores.

- En la quinta prueba, vamos a verificar que la interfaz **New Comment**, es capaz de detectar que su único campo está vació.

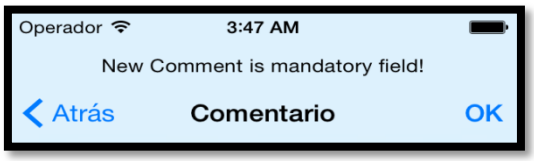
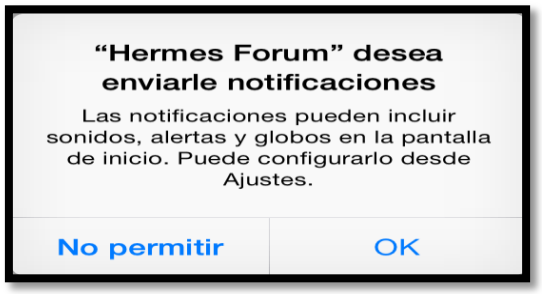


Ilustración 67. Pruebas en la pantalla de Comentario.

La prueba realizada fue un éxito.

- En la sexta prueba, comprobamos si al elegir "No permitir" en aceptar el envió de notificaciones, de verdad no se recibirá ninguna.



Efectivamente, no recibimos ninguna notificación, habiendo enviado varios mensajes en el chat de grupo para comprobarlo. Pero por desgracia, encontramos otro error en el proceso, que podría ser fatal para una de las funcionalidades principales de la aplicación.

Como hemos comentado anteriormente, en el primer tutorial realizado para crear nuestra interfaz principal, uno de los pasos era desactivar el Auto Layout de nuestro **Main.storyboard**.

En un principio pensé que las APIs ofrecidas por C2Call solo eran compatibles con iPhone 5, porque si usabas cualquier otro iPhone, el tamaño de los botones y demás características terminaba deformado. Al desactivar el Auto Layout, siguiendo los pasos del tutorial, quitamos todos los contenedores creados inicialmente por C2Call para compatibilizar sus APIs con cualquier tipo de iPhone.

Este error provoco que el **Text View** del chat dejara de autoincrementarse, mostrando solo una línea cuando empiezas a escribir y mostrando también solo una cuando se envía al chat.

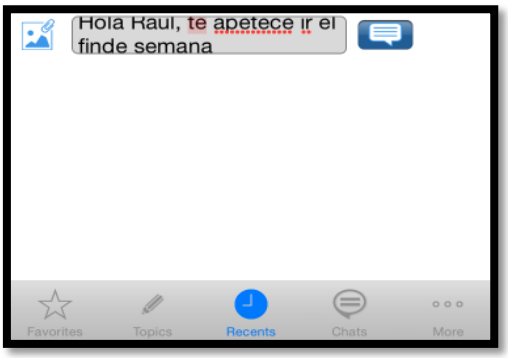


Ilustración 68. Chat sin autoincremento.

Para solucionar este problema, fue necesario reintroducir todas las APIs de C2Call usadas en nuestra aplicación y volver a compatibilizar las interfaces creadas por nosotros, usando esta vez los contenedores.

Pruebas de Rendimiento

La mayoría de las APIs utilizadas en la aplicación son muy sencillas y apenas necesitan recursos para su funcionamiento. Sin embargo la interfaz de Topic y Comment, necesitan acceder a internet y utilizar varias funciones para tratar y mostrar los datos.

Teniendo en cuenta que la información de un foro está creciendo continuamente, se decidió encontrar la cantidad de información que se podía descargar y mostrar a la vez sin ralentizar la aplicación. Utilizando después esos datos, para poner un límite en la descarga y optimizar el tiempo de respuesta, ya que ningún usuario podrá ver todos los temas o comentarios a la vez.

Se ha realizado 2 pruebas básicas para determinar el rendimiento de las interfaces:

- En la primera prueba, comprobamos el funcionamiento de la app al conectarse al servidor de Parse y descargar todos los temas.
    - Introducimos hasta 50 temas en la tabla, la aplicación los descargo y mostró en un segundo.
    - Introducimos en la tabla 100 temas, la aplicación los descargo y mostró en apenas segundo y medio.

- En la segunda prueba, comprobamos el funcionamiento de la app al conectarse al servidor de Parse y descargar todos los comentarios. Esta tabla es mucho más grande que la de temas y necesita también más tratamiento de datos.

    - Introducimos en la tabla 50 comentarios, la aplicación los descargo y mostró en casi 3 segundos.

Con estos datos, se decidió limitar la descarga de temas a 50 y la de comentarios a 20.

Utilizamos las herramientas de xcode para comprobar el rendimiento de la aplicación.

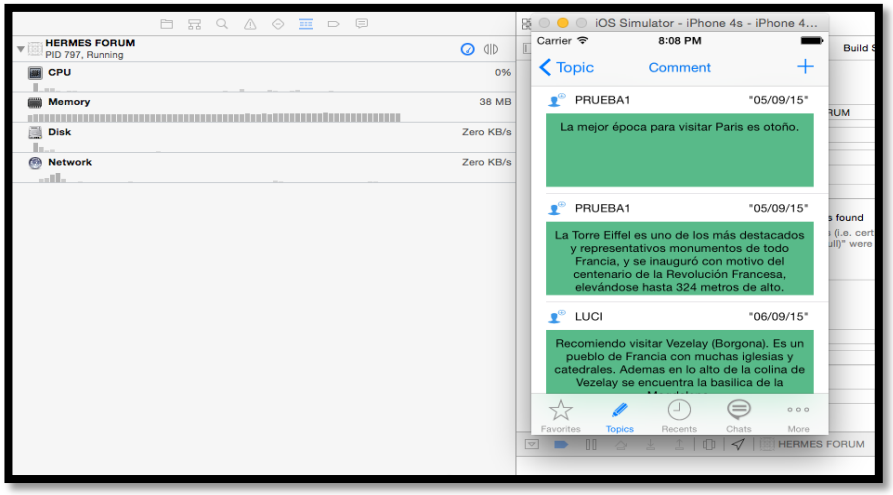


Ilustración 69. Herramienta de optimización.

Pruebas de Instalación

Para poder compilar la aplicación en xcode, es necesario tener un ID de programador de Apple, por lo tanto, no se pudo realizar las pruebas de este apartado.

6. MANUAL DE USUARIO

Autor del documento

José Antonio Murillo Martín. Alumno de la Universidad de Alcalá.

Datos de contacto

E-Mail: akira_mostoles@hotmail.com
Página Web: www.uah.es
Teléfono: 918856505

Versión del documento

1.0
Fecha: 24-8-2015

Licencia del documento

Aviso legal

1. Introducción

La aplicación Hermes Forum nació con la idea de desarrollar un sencillo gestor de temas de viajes, donde se pudiera conocer las experiencias de otros usuarios, pedir opiniones y consejos, buscar amigos, conocer algunos nuevos y sobre todo, ayudar con todos aquellos problemas que pueden surgir al realizar un viaje. Todo ello de una forma sencilla e intuitiva, con una aplicación que nos permite chatear, subir fotos o vídeos, hacer llamadas VoIP, vídeo llamadas en grupo, y mucho más.

El objetivo del presente manual es explicar de forma resumida y en un lenguaje sencillo, los pasos a seguir para utilizar la aplicación "Hermes Forum". La aplicación ha sido desarrollada utilizando el SDK de C2Call y Parse, lo que permite al usuario conectarse de manera normal a través de servidores de seguridad.
Esta conexión evita los bloqueos que puedan surgir a causa de un firewall o router, lo que permite utilizar todos los servicios de la aplicación de forma rápida.

Para garantizar el correcto funcionamiento de Hermes Forum, es necesario que el iPhone del usuario cumpla con los siguientes requisitos:

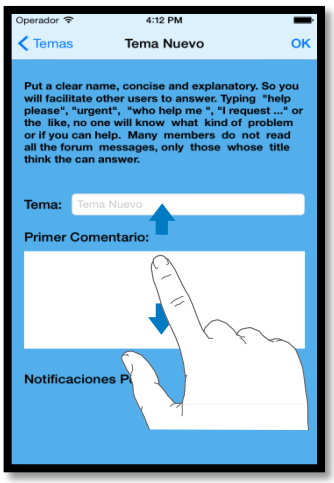
- Se requiere la versión 8.4 o superior de iOS.
- iPhone 4s o superior.
- Conexión a Internet.

Nota: Las apps y servicios que envían o reciben datos a través de una red de telefonía móvil pueden conllevar un gasto adicional. Póngase en contacto con su operador para solicitar información sobre las tarifas y planes de servicios para el iPhone.

En este manual se describe el funcionamiento de Hermes Forum para:

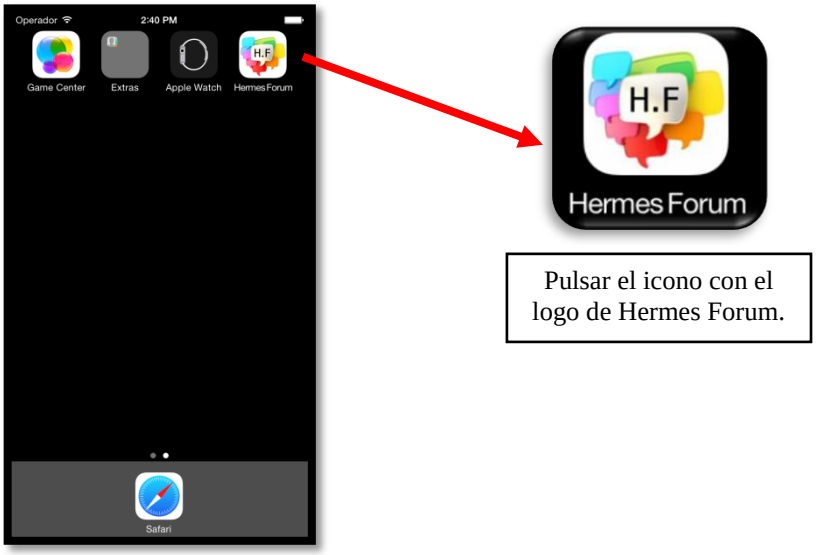
- iPhone 6
- iPhone 6 Plus
- iPhone 5s
- iPhone 5
- iPhone 4s

IMPORTANTE:
Si utilizas un **iPhone 4s**, la pantalla es demasiado pequeña para mostrar correctamente todo el contenido. Pulsa la pantalla y mueve el dedo arriba o abajo para desplazarla.



2. Acceso a la aplicación

Una vez instalada la app en el dispositivo, la podremos encontrar en la sección de Aplicaciones, para iniciarla hay que dar un toque sobre el icono.

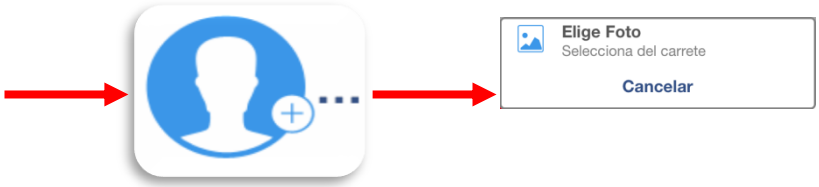


Pulsar el icono con el logo de Hermes Forum.

La aplicación arranca con un sencillo asistente, donde se podrá acceder a **Registrarse** o **Iniciar Sesión**. Si el usuario ya está registrado la aplicación será accesible y entrará directamente a la pantalla de **Favoritos**.

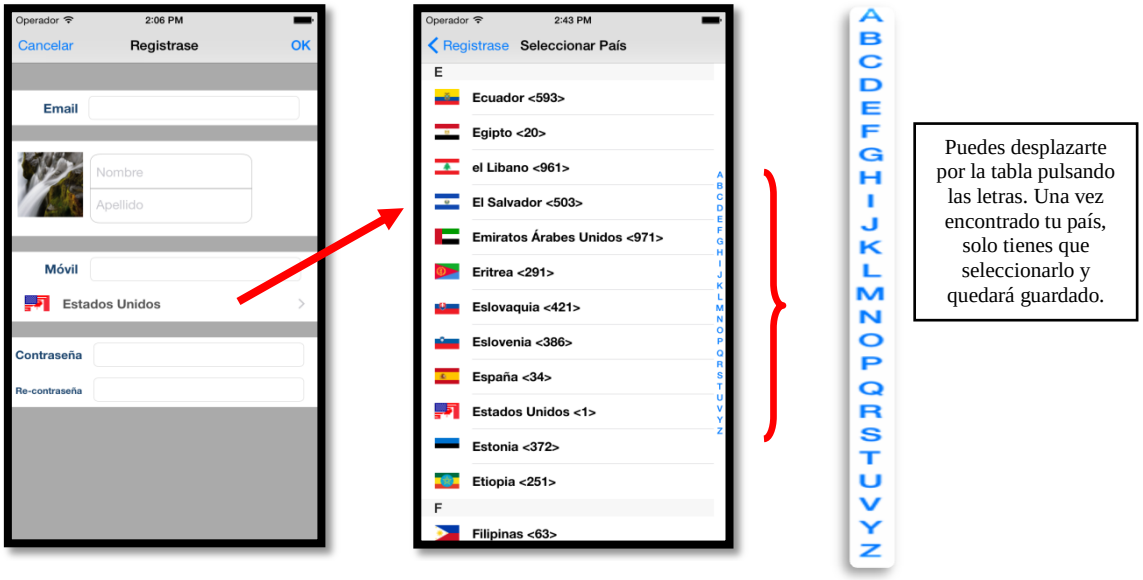
Una vez dentro, tendrás que introducir tu email, usuario, nombre, móvil, ciudad y contraseña. ***Todos los campos son obligatorios***.

Si quieres añadir una foto a tu perfil, ve hacia la parte donde está tu nombre y pulsa el icono.

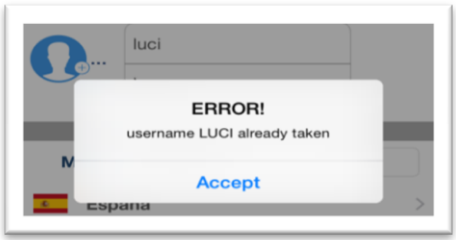


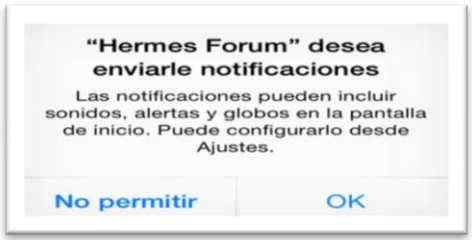
Tendrás la opción de elegir la foto entre las diferentes carpetas configuradas en tu dispositivo.

Después esta la opción de seleccionar tu país de residencia, donde se te asignará el código de prefijo necesario para realizar llamadas internacionales en la versión avanzada.



Recuerda que todos los campos de la ventana registro son obligatorios, una vez completado tu registro pulsa el botón **Ok**. Si ya está registrado tu nombre de usuario o email, la aplicación te mostrará un mensaje de error, en caso contrario, serás redirigido a la pantalla de Favoritos y saldrá un mensaje para aceptar las notificaciones.





Si ya estás registrado, pero te desconectaste la última vez que usaste Hermes Forum, tienes la opción de **Iniciar sesión** en la pantalla de inicio.

Los tres campos de la ventana de **Login** son obligatorios. Al utilizar los servicios de **C2Call** y **Parse**, es necesario introducir el correo registrado para poder usar las principales características de la aplicación.

Si no recuerdas la contraseña, la aplicación incorpora un sistema de seguridad para recuperar la clave. Pulsa el botón **¿Olvidaste tu contraseña?** y accederás a la pantalla correspondiente.

En esta pantalla solo es necesario introducir el correo electrónico, **C2Call** enviará un email con los pasos a seguir para restablecer la contraseña.

Cuando tu email esté introducido, pulse el botón para recibir el correo.

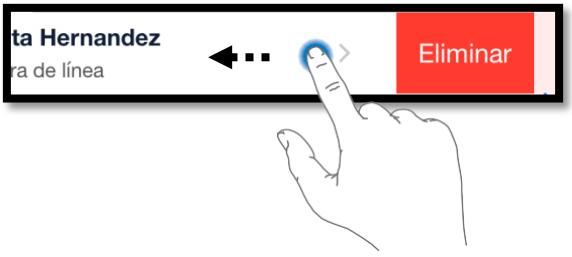
3. Pantalla principal

Si ya estás registrado en Hermes Forum, una vez iniciada la aplicación entrará directamente a la ventana de **Favoritos**.

1) Los estudios demuestran que la mitad de los usuarios van directos al botón de búsqueda, sin echar una ojeada al contenido. Con este dato, el buscador se convierte en una herramienta de primer orden en la estructura de una agenda.
2) Chatear con amigos y conocer algunos nuevos son dos de las competencias principales de Hermes Forum, donde la agenda se convierte en una función vital.
   La agenda mostrará si el amigo está conectado o fuera de línea en todo momento, los grupos de chat creados y la opción avanzada CallMe Link.

CallMe-Link es un método patentado por la compañía **C2Call**, que permite a los usuarios conectarse instantáneamente a través de internet, haciendo posible las llamadas VoIP, vídeo llamadas y mucho más.

IMPORTANTE:
Puedes borrar a un amigo de la agenda deslizando su nombre hacia la izquierda con el dedo.

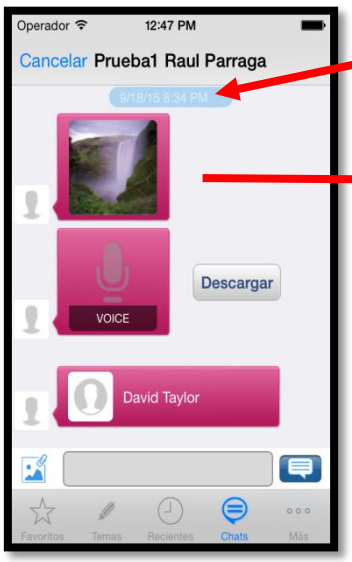


3.1 Comandos del chat

Dentro de los comandos del chat tienes dos botones:

El primero te permite subir un mensaje al chat. **IMPORTANTE:** ***Escribir todo en mayúsculas se considera gritar y dificulta la lectura***.

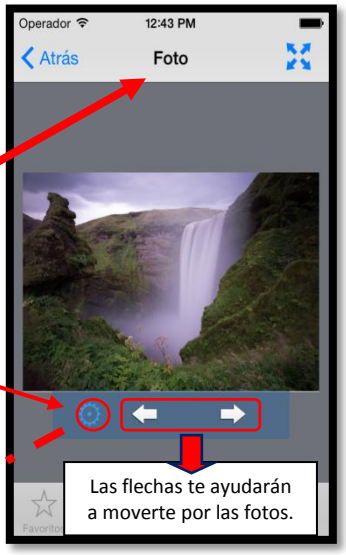
El segundo te permite acceder al menú de opciones, donde podrás subir fotos o vídeos, enviar correos de voz, etc.

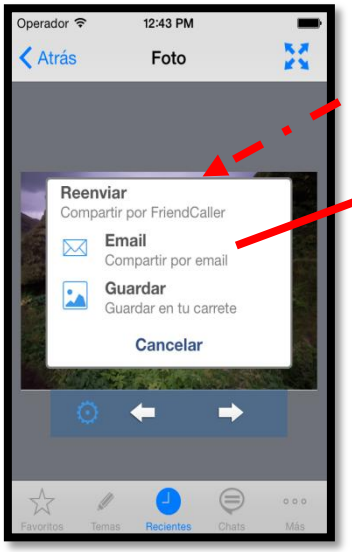


Así se verán los mensajes desde el punto de vista del receptor.

Si pulsas la imagen, pasarás a la pantalla de **Foto**. Donde podrás ver todas las fotos enviadas en este chat.

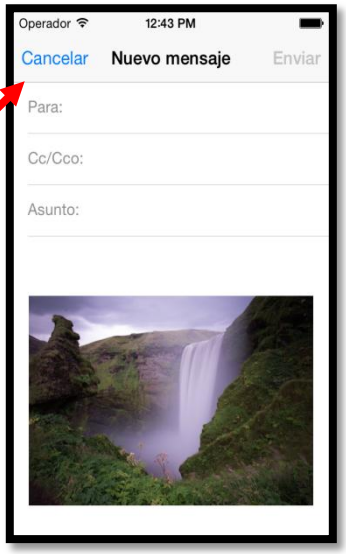
Dentro de esta pantalla, hallarás un botón de configuración. Si lo pulsas, te mostrará las opciones de guardar o enviar por email.





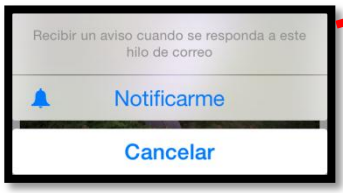
Si pulsas la opción de **Email**, te mostrará la pantalla de configuración de mensajes.

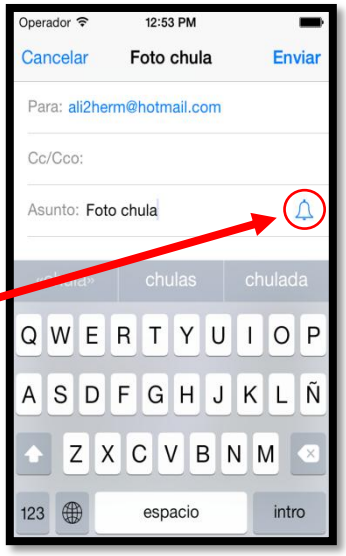
Podrás poner una dirección de correo o elegir una de tu agenda. También se puede añadir un mensaje a la foto.

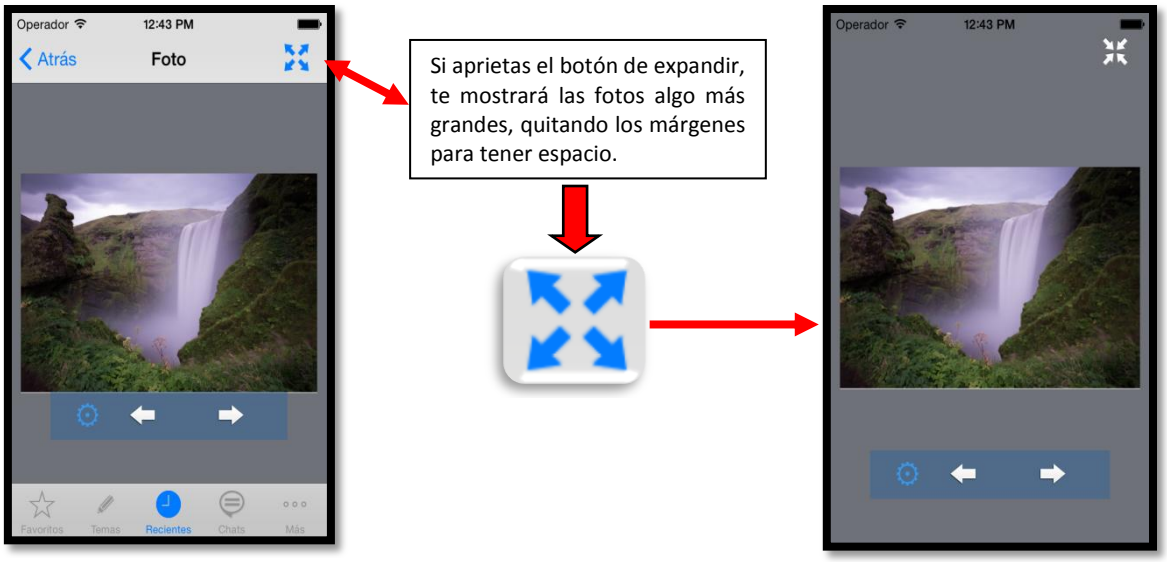


Una vez terminado de rellenar, pulsa el botón **Enviar**. Si prefieres guardarlo, pulsa **Cancelar**.

Si pones un **Asunto**, tendrás la opción de recibir una notificación cuando se responda.







Las imágenes que recibes por chat se descargan de forma automática para que tus últimas fotos estén disponibles rápidamente, las imágenes descargadas aparecerán en tu galería.

Para los archivos multimedia como vídeo y audio, te recomendamos que hagas un archivo o carpeta dentro de la carpeta de imágenes.

4. Foro

Para entrar en el Foro, solo tienes que pulsar el icono de **Temas**, situado en el panel de control.

En la cabecera de la pantalla de **Temas** podemos encontrar dos botones:

Para evitar que la aplicación esté continuamente accediendo al servidor de **Parse**, se descargarán todos los foros la primera vez que se inicie la pantalla de **Temas**. Cuando recibas la notificación de tema creado, pulsa el botón y se actualizará la pantalla.

Para mirar los comentarios, pulsa el tema deseado y entrarás en la pantalla adecuada de **Comentarios**, donde se encuentra únicamente los comentarios relacionados con dicho tema.

Para dejar un comentario pulsa el botón y entrarás en la pantalla de **Comentario.**

Una vez terminado el comentario pulsa el botón **OK** para enviarlo. Si no deseas que alguien reciba la notificación de comentario creado, desplaza el botón de notificaciones hacia la izquierda.

Al igual que en la pantalla de **Temas**, para evitar que la aplicación este continuamente accediendo al servidor de **Parse**, se descargarán todos los comentarios una sola vez. Si quieres actualizar la pantalla pulsa el título **Comentarios**.

Los comentarios están *ordenados por fecha de creación* y se mostrarán en orden descendente, siendo el primero el comentario más antiguo.

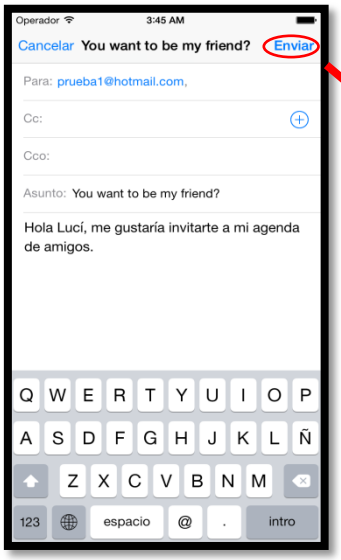
IMPORTANTE:
No existe límite de tamaño al escribir un comentario, pero las ventanas de color verde en la pantalla de **Comentarios** tienen una dimensión fija y será necesario desplazar el mensaje.
Pulsa el mensaje y mueve el dedo arriba o abajo para desplazarlo.

4.1 Invitar a un desconocido

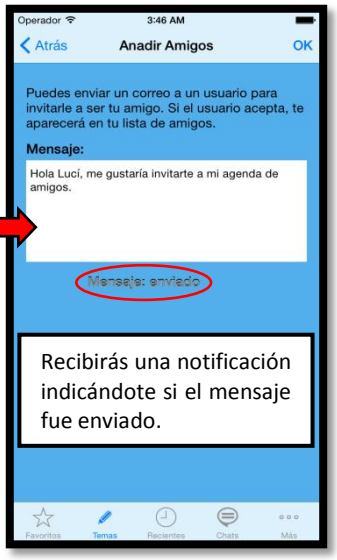
Hermes Forum ofrece la posibilidad de invitar a un desconocido a tu agenda de amigos. Cuando estés en la pantalla de **Comentarios**, puedes invitar a cualquier usuario pulsando el icono a la izquierda de su nombre.

Si pulsas el botón te reenviará a la pantalla de **Añadir Amigos**.
Si deseas puedes escribir un mensaje personalizado o pulsar directamente el botón **OK**.
IMPORTANTE: ***Se enviará un correo predefinido si el campo mensaje está vacío.***

Si el campo correspondiente al mensaje no está vacío, al pulsar el botón **Ok**, te mostrará la pantalla de **configuración de mensajes**.



Si no deseas hacer más modificaciones en el correo, pulsa **Enviar**.
Si por el contrario decidiste no enviarlo pulsa **Cancelar**.



Recibirás una notificación indicándote si el mensaje fue enviado.

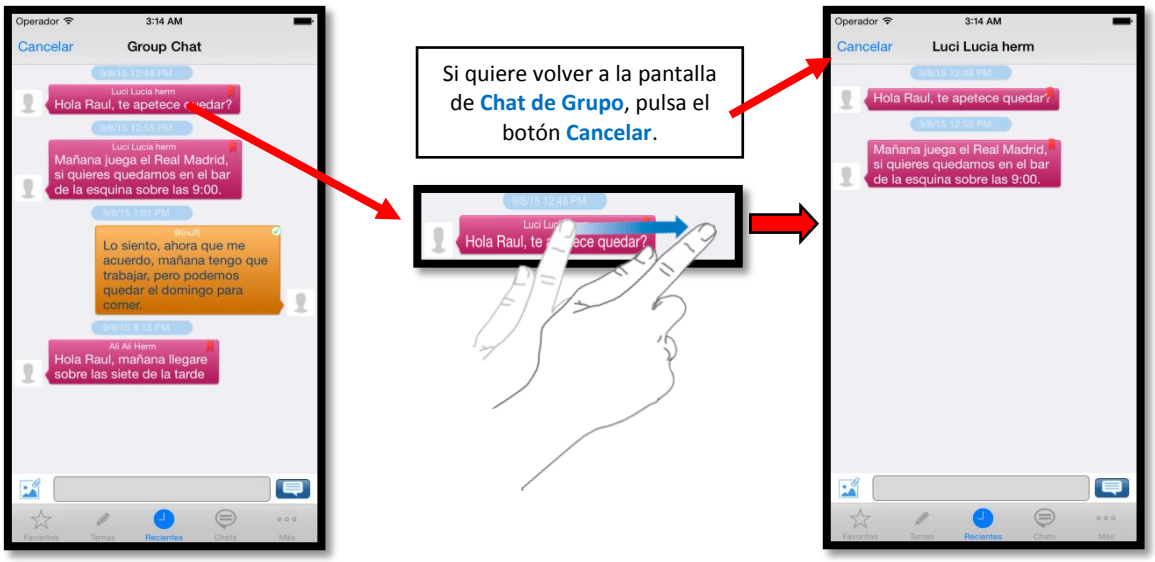
El usuario *recibirá el mensaje en su correo electrónico*, de esta forma evitamos posibles errores si el destinatario está desconectado de la aplicación.

El destinatario solo necesita introducir tu dirección de correo en la opción de Encontrar Amigos que se explicará más adelante.

5. Recientes

Cuando la aplicación está conectada a Internet, el servidor de C2Call comprueba si existe algún nuevo mensaje. Para mirar los nuevos mensajes solo tienes que pulsar el icono de **Recientes**, situado en el panel de control.

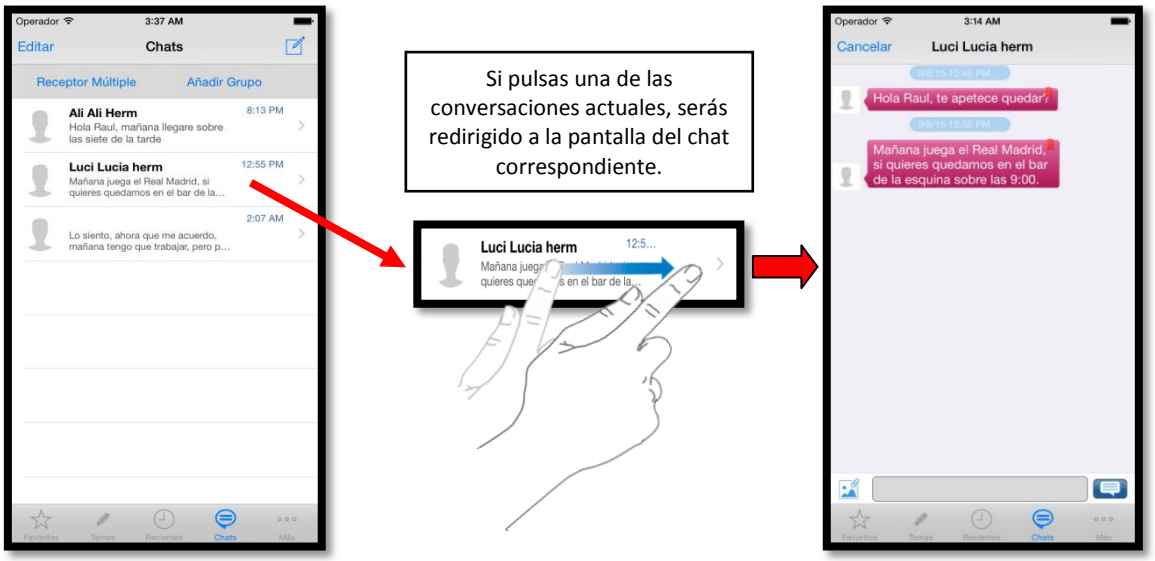
La pantalla de **Chat de Grupo** mostrará los últimos mensajes enviados en cada chat o grupo que tengas abierto, si quieres entrar en uno de los chats, pulsa el mensaje correspondiente y serás reenviado al chat particular.



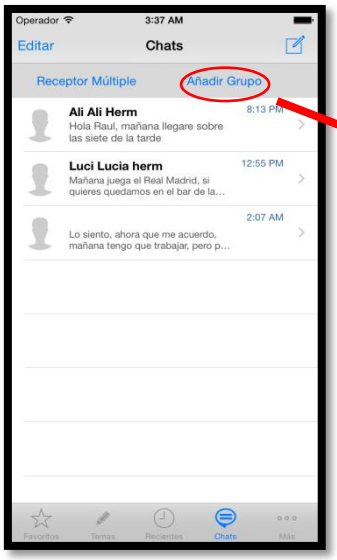
6. Chats

Si quieres saber qué conversaciones tienes abiertas y con quién, solo tienes que pulsar el icono de **Chats**, situado en el panel de control.

La pantalla de **Chats** te mostrará todas las conversaciones actuales, los grupos creados y la fecha y hora en que se crearon.

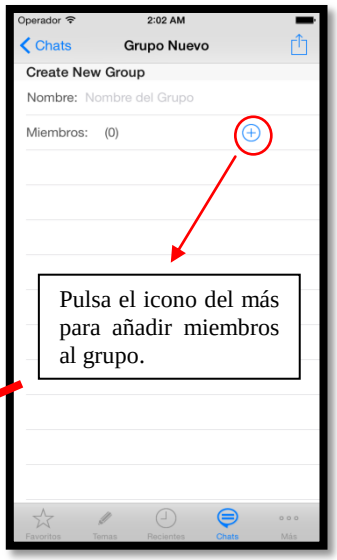


6.1 Crear y editar grupos

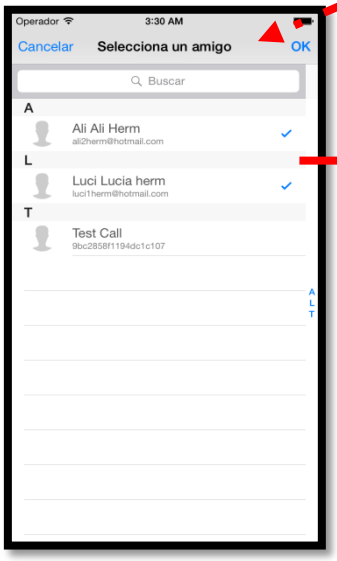


Añadir Grupo

Si pulsas en **Añadir Grupo**, serás reenviado a la pantalla de **Grupo Nuevo**, donde podrás crear un grupo de amigos.

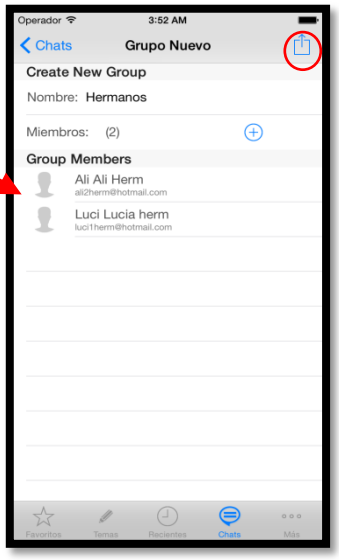


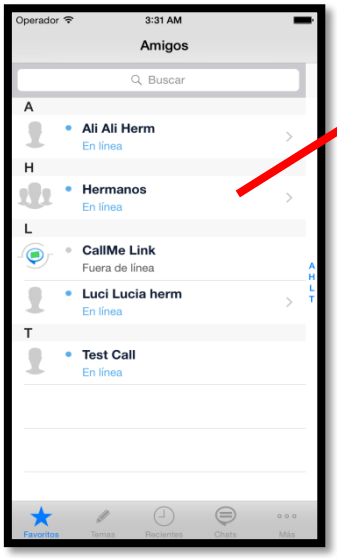
Pulsa el icono del más para añadir miembros al grupo.



No existe límite en el número de miembros. Selecciona los compañeros deseados y pulsa **OK**, si ya no deseas añadir más compañeros pulsa **Cancelar**.

Si quieres guardar los datos del grupo, pulsa el icono a la derecha del título.

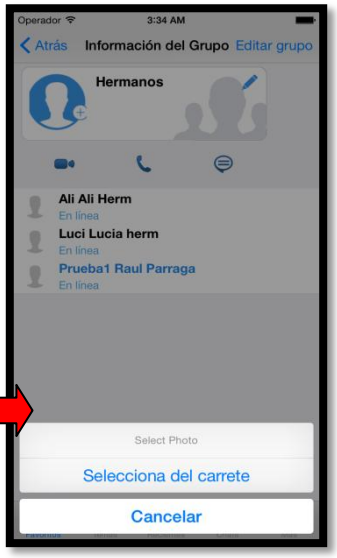




Una vez guardado, aparecerá en la pantalla de **Favoritos**.
Si pulsas en el nombre, entrarás en la pantalla de **Información de Grupo**, donde podrás modificar su nombre y añadir una imagen.

Editar nombre.

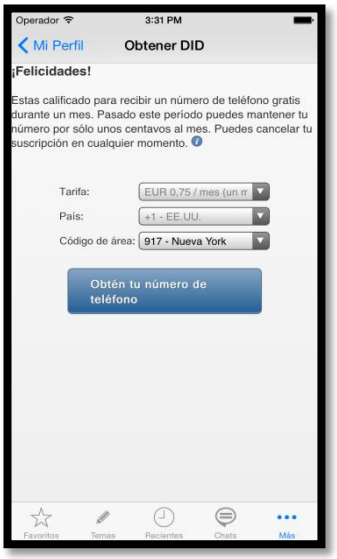
Añadir imagen.



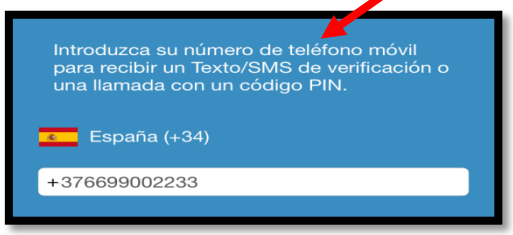
7. Más

Para entrar en el menú de opciones de la aplicación, solo tienes que pulsar el icono de **Más**, situado en el panel de control.

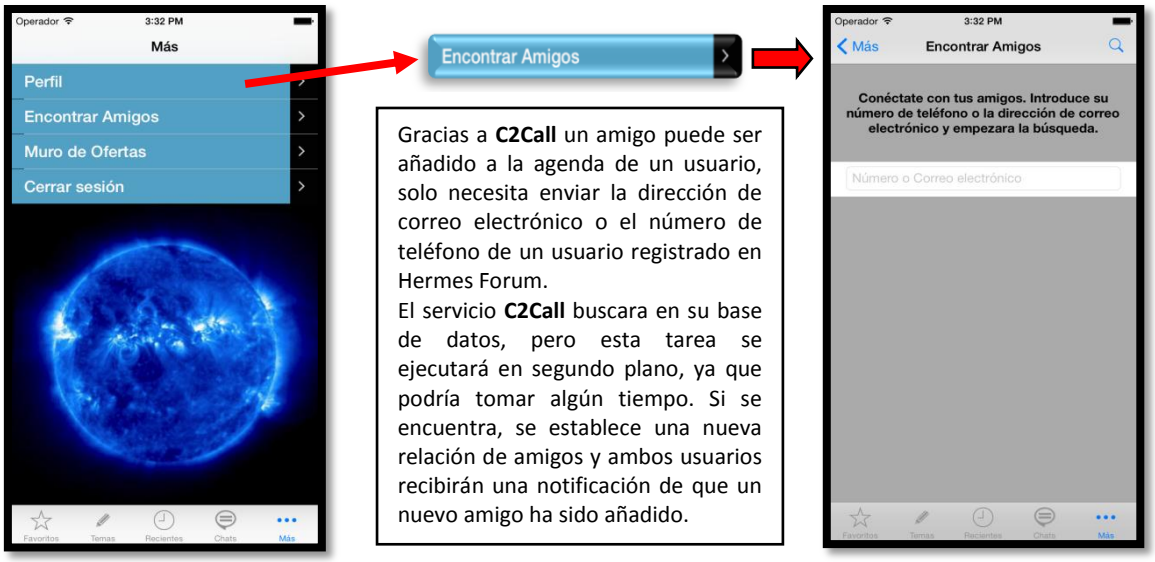
Si pulsas en **Perfil**, te mostrará la pantalla de **Mi Perfil**, donde podrás ver tus datos personales, el crédito actual de tu cuenta y la opción de pedir un DID (número telefónico virtual)



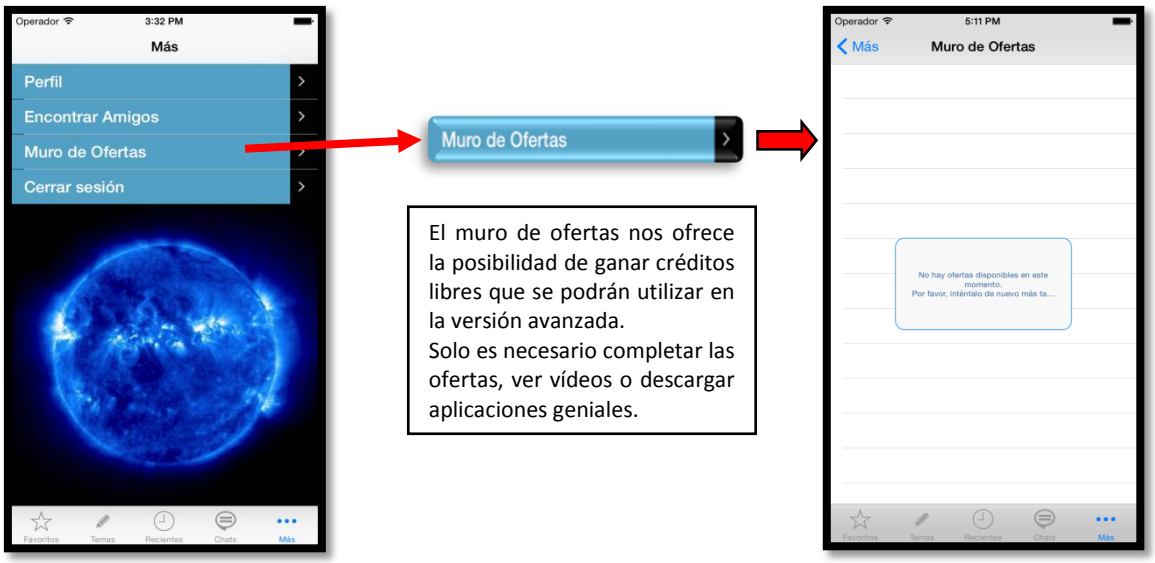
Un número telefónico virtual, conocido como **DID**, es un número local que puede pertenecer a cualquier ciudad del mundo, desde el cual una persona puede llamar a teléfonos fijos y móviles.
El **DID** tiene un cargo fijo mensual por el servicio, pero gracias a los créditos libres de **C2Call** puedes conseguir que sea gratis.
El número está asociado a tu cuenta VoIP, la cual puede ser usada solo en la versión avanzada.



C2Call ofrece un servicio de identificación de llamadas, pero antes es necesario verificar el número de teléfono.
IMPORTANTE: ***Ten cuidado al introducir tu código de prefijo.***



Gracias a **C2Call** un amigo puede ser añadido a la agenda de un usuario, solo necesita enviar la dirección de correo electrónico o el número de teléfono de un usuario registrado en Hermes Forum.
El servicio **C2Call** buscara en su base de datos, pero esta tarea se ejecutará en segundo plano, ya que podría tomar algún tiempo. Si se encuentra, se establece una nueva relación de amigos y ambos usuarios recibirán una notificación de que un nuevo amigo ha sido añadido.



El muro de ofertas nos ofrece la posibilidad de ganar créditos libres que se podrán utilizar en la versión avanzada.
Solo es necesario completar las ofertas, ver vídeos o descargar aplicaciones geniales.

Botón de inicio
Si cierras la aplicación no recibirás notificaciones, se recomienda usar el botón de inicio.

7. CONCLUSIONES

El turismo es uno de los pilares de la economía mundial y gracias al auge del turismo en Europa, aumenta la importancia de la información. Mientras mejor informados estemos, menos sorpresas encontraremos al realizar nuestro viaje.

Hasta ahora, en iPhone solo había aplicaciones que te permitían ver foros e interactuar con ellos a través de safari, pero no existía una aplicación que te permitiera crear tu propio foro, donde poder dar consejos, hacer amigos, chatear, realizar llamadas y mucho más, consiguiendo una comunicación fluida y sencilla a través de Internet.

Este proyecto aporta el concepto de **red social** a los foros, de esta forma, aumentamos la calidad de la información obtenida por los usuarios, no es sólo dejar comentarios o consejos, sino interactuar con amigos y desconocidos, siendo una aplicación donde los usuarios son los que deciden el tipo de información que quieren compartir.

De esta forma Hermes Forum se convierte en una red social de viajeros basada en la transmisión de información, que establecerá una nueva tendencia en los viajes y se perfilará como plataforma indispensable para la optimización de la distribución y el consumo de nuevos servicios.

Tras la finalización de la aplicación y la conclusión de las pruebas, se puede determinar que los objetivos que se propusieron al inicio de este proyecto se han cumplido en su totalidad, consiguiendo desarrollar una aplicación sencilla e intuitiva, que facilitará su manejo al usuario final.

8. TRABAJOS FUTUROS

Aunque el proyecto ha finalizado satisfactoriamente, a medida que diseñaba la interfaz principal y se desarrollaban los métodos de la aplicación, iban surgiendo nuevas ideas. Algunas de estas pudieron ser planificadas y desarrolladas, pero por falta de tiempo y conocimientos muchas no pudieron ser implementadas en el proyecto.

Estas funcionalidades son las que a continuación se detallan:

- **Mejoras en el diseño**

Por falta de tiempo y conocimientos, muchas partes de la app no fueron diseñadas de la forma deseada en un principio. Si bien la Aplicación "funciona" y las funcionalidades están implementadas para que el usuario comprenda su funcionamiento de forma intuitiva, para el desarrollo de la interfaz secundaria se utilizaron solo componentes nativos del SDK de iOS, siendo esta parte visualmente poco atractiva.

- **Aumentar los idiomas de la aplicación**

Por falta de tiempo se utilizó solo un idioma base (Ingles) y como segundo idioma el español, pero para un foro de viajes es una buena práctica hacerlo en varios idiomas o dialectos. Además de ser un detalle que el usuario agradecerá, permitirá ampliar el rango de mercado de nuestra aplicación.

- **Traductor automático**

Los idiomas son en multitud de ocasiones una barrera difícil de salvar para la comunicación entre dos personas. Por eso me hubiera gustado añadir un traductor automático y ayudar a los usuarios a poder comunicarse libremente.

- **Adaptación a iPad**

Debido a las diferencias evidentes de tamaño entre un iPhone y un iPad, muchas guías de desarrolladores recomiendan diseñar la app exclusivamente para cada plataforma, de esta forma se podrá usar las herramientas específicas para iPad que mejoran la experiencia de usuario y la usabilidad. Siguiendo este consejo, se decidió diseñar la app exclusivamente para iPhone, pero por cómo avanza el mercado de tablets y teniendo en cuenta los nuevos modelos de iPhone, es imprescindible adaptar la aplicación para este tipo de dispositivos.

- **Optimización de las tablas de Parse**

Al no tener un conocimiento demasiado extenso sobre el uso de Parse, algunas entradas o datos fueron duplicados en diferentes tablas. Se podría optimizar las tablas

usando relaciones y punteros, reduciendo también el espacio ocupado y posibilitando que las consultas fueran más rápidas y eficientes.

- **Mejor uso de las notificaciones push**

Gracias al SDK de **C2Call** la mayoría de las notificaciones son enviadas a través de sus servidores, permitiendo ahorrar mucho tiempo y esfuerzo. Pero tiene un defecto de base, no puedes modificarlas o crear una nueva.
Este problema se podría solucionar muy fácilmente registrando los dispositivos de nuestros usuarios en Parse, para permitirnos posteriormente enviarles notificaciones push tanto desde código, como desde el panel de Parse, pero para usar este servicio debes tener cuenta de desarrollador.

- **Mejoras de rendimiento**

La aplicación está continuamente accediendo al servidor de Parse para realizar diversas consultas, pero debido a las limitaciones del diseño, los datos tienen que ser tratados para su representación visual. Podría mejorarse el rendimiento limitando la información obtenida, devolviendo solo las columnas o filas necesarias.
En un principio no fui capaz de encontrar una forma más eficiente, pero estoy seguro de que dentro de la extensa documentación de Parse tiene que existir un método más eficaz.

- **Integración con Facebook**

Parse ofrece el SDK Facebook, que se puede integrar con Parse para poder vincular fácilmente a los usuarios con sus identidades de Facebook y de esta forma autentificar el email usado para el registro de la aplicación.

- **Integración de nuevos sectores**

La aplicación se centra inicialmente en el sector turístico, por ser el que más se ha incrementado en los últimos años, ofreciendo una mayor demanda de servicios e información. Pero a medida que aumenta la cantidad de usuarios, se podrá añadir otros sectores como el deporte, las finanzas, la educación, etc. De este modo la aplicación tendrá infinidad de posibilidades futuras.

ANEXO 1 Presupuesto

En este anexo se recoge la estimación económica del proyecto. Para ello se tendrá en cuenta el número de horas trabajadas, la adquisición de licencias, el material utilizado y demás costes asociados.

El número total de horas utilizadas en el proyecto fueron 549, las cuales se pueden dividir en 11 etapas, con sus respectivas tareas:

- **Etapa 1: 16 horas**, para definir el proyecto (finalidad de la app, funcionalidad básica, etc.)
- **Etapa 2: 25 horas**, para la instalación de la máquina virtual y los programas necesarios para el desarrollo del proyecto (Xcode, Prepo e iMage Tools). No se pudo instalar Yosemite directamente, se instaló un sistema operativo anterior y después se actualizo a la última versión. **Desglose: 8 horas** en la búsqueda de información, **7 horas** para descargar e instalar la máquina virtual, **4 horas** en actualización a Yosemite, **1 hora** para configurar el sistema operativo y **5 horas** en la instalación de Xcode, Prepo e iMage.
- **Etapa 3: 38 horas**, para la realización de dos cursos básico (uno de objective-c y otro de Xcode). Además de la realización de pequeñas pruebas para fortalecer los conocimientos adquiridos en los cursos.
- **Etapa 4: 80 horas**, para el desarrollo de la primera parte del proyecto, usando C2Call en el diseño de la interfaz principal. **Desglose: 16 horas** en la búsqueda de información, **24 horas** en el diseño e integración de la interfaz principal, **19 horas** en personalizar diversas interfaces de usuario y **21 horas** en la solución de varios warnings producidos por el SDK de C2Call.
- **Etapa 5: 50 horas** en diseñar la interfaz secundaria que ofrecerá las opciones de foro. **Desglose: 21 horas** en la búsqueda de información y **29 horas** en el diseño de la interfaz secundaria.
- **Etapa 6: 100 horas** para el desarrollo de la segunda parte del proyecto, usando los servicios de Parse e integrándolos en el diseño principal de C2Call. **Desglose: 18 horas** en la búsqueda de información, **16 horas** para rediseñar y mejorar la interfaz de Login/Registro, **54 horas** en la creación de métodos y funciones, que se encargarán del tratamiento de datos y su representación visual dentro de la interfaz secundaria y **12 horas** en el diseño de las tablas y protocolos de Parse.
- **Etapa 7: 20 horas** para compatibilizar el diseño con los diferentes iPhone.
- **Etapa 8: 12 horas** para la traducción de la aplicación a un segundo idioma.
- **Etapa 9: 36 horas** en la realización de pruebas.
- **Etapa 10: 28 horas** en solución de errores.
- **Etapa 11: 144 horas** dedicadas a la documentación.

Suponiendo que un solo programador junior se encargue de todo el desarrollo de la aplicación (análisis, diseño y pruebas) y tomando el salario de un programador junior a 8 € la hora, se puede concluir que el gasto del proyecto perteneciente a costes de personal es de **4.392 €**.

El coste de las licencias de software se recoge en la siguiente tabla:

Software	Coste
OS X Yosemite	Gratuito
Xcode	Gratuito
Licencia de desarrollador Apple	99 €
VMware Workstation 11	149 $ ± 132,55 €
Prepo	Gratuito
iMage	Gratuito
	231,55 €

Tabla 4. Costes de Software.

Parse es una plataforma de pago, pero cuenta con una capa gratuita de dimensiones bastante considerables. Concretamente, con la cuenta gratuita tenemos derecho sin ningún coste, a disfrutar de las siguientes funcionalidades al mes:

- 1.000.000 peticiones API.
- 1.000.000 notificaciones push.
- 1 Gb de espacio en disco.
- Un máximo de 20 peticiones por segundo.

Para la mayoría de aplicaciones de pequeña o media escala, esta capa gratuita será más que suficiente. Sin embargo, si necesitamos más, tenemos una cuenta **Pro** con mucha más capacidad por **199$/mes**, e incluso una cuenta **Enterprise** adaptada a cada caso empresarial.

Al igual que Parse, C2Call es una plataforma de pago que cuenta con varios servicios de forma gratuita. Estos servicios son:

- Garantiza todas las funcionalidades gratuitas a 1 App.
- Uso ilimitado de Vídeo Chat, llamadas de voz y mensajería instantánea.
- Uso ilimitado de las notificaciones push.
- Activación mensual de hasta 10.000 usuarios.
- Anuncios gratis para el muro de ofertas.
- Buscador de amigos automático.
- Acceso al foro de la comunidad, donde el soporte técnico responderá a todas tus preguntas.

Si la aplicación tuviera éxito y fuera necesario más servicios, también cuenta con una cuenta **Pro** con mucha más capacidad por **99$/mes**, e incluso una cuenta **Enterprise** adaptada a cada caso empresarial.

En coste de los componentes hardware para la realización de este proyecto son:

Hardware	Coste
Lenovo G510	600 €
Memoria RAM 8gb (2x 4gb)	59,99 €
	659,99 €

Tabla 5. Costes de Hardware.

Así pues, el coste total del proyecto asciende a **5.283,54 €**, como se refleja en la siguiente tabla:

Apartado	Coste
Personal	4.392,00 €
Software	231,55 €
Hardware	659,99 €
	5.283,54 €

Tabla 6. Coste total del proyecto.

ANEXO 2 Planificación

Una de las partes más importantes en un proyecto es la organización temporal de las tareas que se deben realizar para llevarlo a cabo. Para crear una gran app, hace falta un gran plan.
El proyecto estará dividido en las siguientes tareas:

- **Definir un proyecto**: Antes de elaborar un proyecto, se debe decidir en qué va a consistir exactamente, cuál será su **alcance** (finalidad de tu app, los destinatarios y su forma de utilizar la app, su funcionalidad básica, etc.) y que se espera alcanzar con el proyecto. Duración estimada de la tarea: **16 horas**.
- **Cursos básicos**: Para crear una app para iPhone partiendo de cero, es necesario aprender a usar de forma básica el lenguaje de programación usado por Apple (Objective-C). También será necesario aprender el manejo básico de Xcode, herramienta que nos facilitará todo lo necesario para crear y simular tu app. Duración estimada de la tarea: **38 horas**.
- **Inventario de activos**: Muchas veces, la mejor forma de diseñar una app es aprovechar las tecnologías existentes. Una buena opción puede ser realizar una investigación detallada sobre los posibles servicios web, que pueden plantear una solución a nuestra funcionalidad básica. Duración estimada de la tarea: **8 horas**.
- **Adquirir recursos y herramientas**: Una vez definido el proyecto y realizado el inventario de activos, procedemos a adquirir las herramientas necesarias para comenzar la app. Esta fase se dará a lo largo de todo el proyecto, en la medida en que los recursos y herramientas (librerías, servicios, documentación, SDK) se requieran en una determinada actividad. Duración estimada de la tarea: **17 horas**.
- **Planificación del proyecto**: El jefe de proyecto determina las tareas, cargas de trabajo, calendario y medios a utilizar. Gracias al SDK de iOS, C2Call y Parse el equipo de desarrollo podrá dedicar menos tiempo a la programación y más al diseño de un entorno ideal para el usuario. Duración estimada de la tarea: **8 horas**.
- **Diseño de la Interfaz Principal**: Para el desarrollo de la primera parte del proyecto, usaremos las APIs de **C2Call**, que nos permitirán diseñar la interfaz principal de la aplicación. También será necesario registrar la aplicación en los servicios de C2Call, para poder disponer de todas las funcionalidades. Duración estimada de la tarea: **72 horas**.
- **Diseño de la Interfaz Secundaria**: La interfaz secundaria se encargará de controlar las características propias del Foro, permitiendo crear temas, añadir comentarios, invitar usuarios, enviar notificaciones, etc. Duración estimada de la tarea: **50 horas**.
- **Diseño de la Base de Datos**: Para el desarrollo de la segunda parte del proyecto, usaremos los servicios de Parse, que nos permitirá crear tablas no-SQL para poder almacenar los datos de Login / Registro y toda la información relacionada con el Foro. También será necesario registrar la aplicación en los

servicios de Parse, para poder disponer de todas sus funcionalidades. Duración estimada de la tarea: **100 horas**.

- **Compatibilización de diseño**: La gran diferencia de tamaño entre los diferentes iPhone (el iPhone 4s tiene 3'5 pulgadas y el iPhone 6 4'7), hace necesario compatibilizar el diseño, evitando el ocultamiento o deformación de algunos botones, textos, etc. Duración estimada de la tarea: **20 horas**.
- **Introducción de segundo idioma**: Se traducirá la aplicación al castellano para tener un mayor impacto en la presentación de la app. Facilitando además un mayor rango de mercado y aceptación por parte de los usuarios. Duración estimada de la tarea: **12 horas**.
- **Pruebas y validación**: En esta tarea se realizarán las pruebas de validación del rendimiento, la optimización de la interfaz, la conexión y la usabilidad en condiciones reales. Duración estimada de la tarea: **36 horas**.
- **Solución de Errores**: Una vez identificados los errores, usaremos los datos obtenidos en las pruebas para corregir rápidamente los problemas. Duración estimada de la tarea: **28 horas**.
- **Documentación**: Se documentarán todas las tareas realizadas durante la ejecución normal del proyecto (herramientas, librerías, diseño, etc.) además de los servicios utilizados como **C2Call** y **Parse** (características principales, ventajas y desventajas) y un Manual de Usuario para explicar el uso correcto de las funcionalidades de la aplicación. Duración estimada de la tarea: **144 horas**.
- **Implantación**: Para empezar el proceso de implantación, tendrás que certificar y distribuir la app a través del programa Enterprise para desarrolladores de iOS y firmar y crear el proyecto en Xcode.

El siguiente diagrama de Gantt muestra la duración estimada de cada uno de las tareas descritas anteriormente. Se puede observar que todas las tareas necesitan que las anteriores hayan sido finalizadas para poder iniciarse.

Esta es la planificación realizada en el mes de Junio de 2015, al inicio del proceso de desarrollo del proyecto.

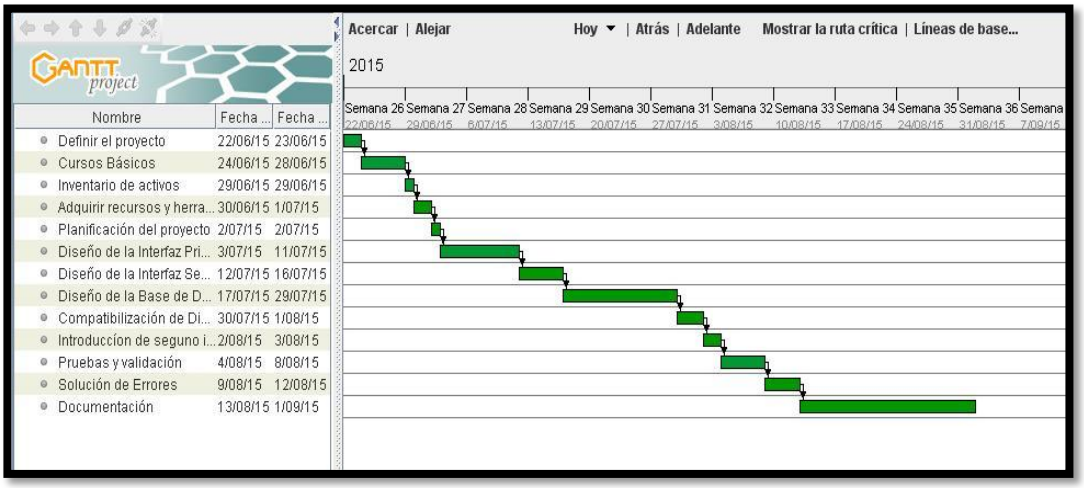


Ilustración 70. Diagrama de Gantt de planificación inicial.

A continuación se muestran el diagrama de Gantt con los periodos de tiempo llevados a cabo durante la elaboración del proyecto:

- Verde: La tarea se terminó antes de tiempo.
- Amarillo: La tarea se terminó en el tiempo previsto.
- Rojo: La tarea se terminó más tarde.

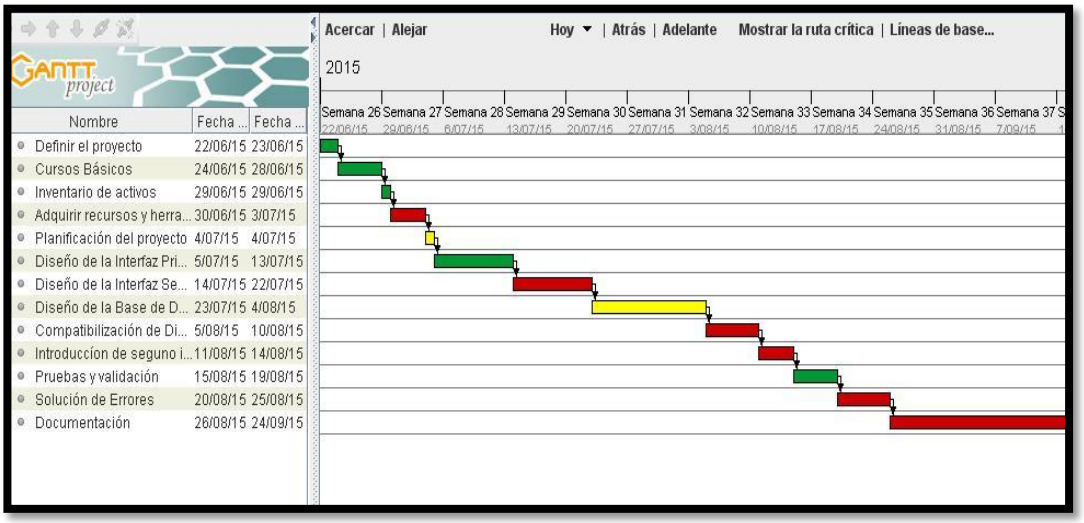


Ilustración 71. Diagrama de Gantt de planificación final.

Cabe destacar que la planificación real es más extensa que la inicial, puesto que hubo que alargar el periodo de varias tareas, dedicando más horas a cada una con el fin de poder entregar correctamente el proyecto.

ANEXO 3 Bibliografía

[1] Oscar González. "Objective-c desde Cero", http://codehero.co/series/objective-c-desde-cero.html

[2] Ricardo Moya. "UITableView en Objective-c" , http://jarroba.com/uitableview-en-objective-c-ejemplo-video/

[3] Ricardo Moya. "Pasar Datos entre ViewControllers usando StoryBoards y Segues", http://jarroba.com/pasar-datos-entre-viewcontrollers-usando-storyboards-y-segues/

[4] Ignacio Oyola. "Cadenas de texto en Objective-c", http://untitled.es/cadenas-string-objective-c/

[5] Ignacio Oyola. "Imágenes Launch iPhone iPad", http://untitled.es/imagenes-launch-iphone-ipad/

[6] Aram Julhakyan, Marcos Ramos Rubio y Pol Cirujeda Santolaria. "#pragma mark en Xcode", http://www.zenbrains.com/blog/2010/04/todo-y-pragma-mark-en-xcode/

[7] Javier Cala Uribe. "Guía iOS: Uso del correo", http://www.maestrosdelweb.com/guia-iphone-e-ipad-uso-del-correo/

[8] Miguel Díaz Rubio. "Serie de cinco artículos para Desarrollo iOS: Parse", http://www.migueldiazrubio.com/2013/06/07/desarrollo-ios-parse-i-introduccion-e-instalacion/#

[9] Documentación C2call. "iOS-Tutorial-Simple-Chat-App" https://www.youtube.com/watch?v=qyp_BFORvIk

[10] Raúl Hernáiz Ortega. "Posicionamiento en interiores mediante tecnología Bluetooth y estimación logarítmica". TFG Universidad de Alcalá. Septiembre de 2013.

[11] DesarrolloWeb.com / EscuelaIT. "Curso de Iniciación en Objective- c", https://www.youtube.com/watch?v=vdvHnmYHze8&index=1&list=PLIcuwIrm4rKfbaB7kueHe7D9gwFTt18Xf

[12] Luis Berganza. "Repositorios Populares", https://github.com/berganza

[13] ComputerHoy.com. "Cómo instalar OS X Mavericks en Windows con VMWare", http://computerhoy.com/paso-a-paso/software/como-instalar-os-x-mavericks-dentro-windows-vmware-8816

[14] Sergio Becerril. "Parse, el Back-end que hará tu vida más fácil", www.cfeapps.com/parse-el-backend-que-debes-conocer/

[15] Manuel López. "Xcode: El entorno de desarrollo integrado de Apple", http://iosmac.es/xcode-el-entorno-de-desarrollo-integrado-de-apple.html

[16] Ernesto García. "Mis primeras impresiones de Objective-C", http://blog.continuum.cl/mis-primeras-impresiones-de-objective-c/

[17] Julio César Fernández. "El dilema del desarrollador Apple", https://applecoding.com/opinion/objective-c-swift-dilema-desarrollador-apple

[18] ITIOX. "Xcode: Localizar o Internacionalizar tu App", http://blog.itiox.com/2013/06/xcode-localizar-o-internacionalizar-tu.html

[19] Sergio Becerril. "Cambiar texto y color en un label con Objective-C y Swift", http://www.cfeapps.com/cambiar-texto-y-color-en-un-label-con-objective-c-y-swift/

[20] David Arias Vázquez. "¿Qué es Objective-C?", http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Interfaces/mac/cocoa3_1.html